

**Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky**

**Optimalizácia smerovacieho procesu,
hľadáním ciest v grafových štruktúrach,
využitím strojového učenia**

Diplomová práca

**Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky**

**Optimalizácia smerovacieho procesu,
hľadaním ciest v grafových štruktúrach,
využitím strojového učenia**

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	doc. Ing. Peter Feciľak, PhD.
Konzultant:	Ing. Erika Abigail Katonová

Košice 2026

Bc. Juraj Jurčišin

Abstrakt v SJ

Prepojenosť dnešného sveta vedie k stále väčšej dôležitosti počítačových sietí. Pre zlepšenie kvality komunikácie v počítačových sieťach, veľké množstvo výskumníkov zameriava svoju prácu na počítačové siete. Jedným smerom výskumu v počítačových sieťach je využitie strojového učenia v smerovacom procese. Táto práca sa venuje analýze existujúcich prístupov k využitiu strojového učenia v smerovacom procese a experimentálne vyhodnocuje niektoré z daných navrhovaných prístupov, konkrétne výber cesty pre smerovanie dát a predikciu váh pridelených jednotlivým linkám. Táto práca taktiež navrhuje nový smerovací prístup, ktorý kombinuje oba predchádzajúce prístupy do jedného smerovacieho prístupu. Pokiaľ vieme, takýto prístup nebol navrhnutý v žiadnom predchádzajúcom výskume v čase písania tejto práce. Všetky tri prístupy sú v tejto práci experimentálne vyhodnocované a porovnávané medzi sebou ako aj s tradičnejším prístupom využívajúcim SPF algoritmus.

Kľúčové slová v SJ

smerovanie, počítačové siete, strojové učenie, hlboké učenie posilňovaním

Abstrakt v AJ

The interconnectedness of today's world leads to an increase of the importance of computer networks. To improve the quality of communication in a computer network, many researchers dedicate their research to computer networks. One such avenue of research is the usage of machine learning in the routing process. This thesis analysis current research on the topic of the incorporation of machine learning into the routing process and experimentally evaluates some of the proposed routing methods, namely the selection of a routing path and the prediction of the link weights. This thesis also proposes a novel routing scheme, which combines both previous routing schemes into a single combined method. To the best of our knowledge, this was not done in any previous research at the time of writing. All the three methods are experimentally evaluated and compared with each other and with the more traditional SPF algorithm.

Kľúčové slová v AJ

routing, computer networks, machine learning, deep reinforcement learning

Bibliografická citácia

JURČIŠIN, Juraj. *Optimalizácia smerovacieho procesu, hľadanie ciest v grafových štruktúrach, využitím strojového učenia*. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2026. 105s. Vedúci práce: doc. Ing. Peter Feciľak, PhD.

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

**ZADANIE
DIPLOMOVEJ PRÁCE**

Študijný odbor: **Informatika**

Študijný program: **Informatika**

Názov práce:

**Optimalizácia smerovacieho procesu, hľadaním ciest v grafových štruktúrach,
využitím strojového učenia**

**Optimization of the Routing Process by Finding Paths in Graph Structures, using
Machine Learning**

Študent: **Bc. Juraj Jurčišín**

Školiteľ: **doc. Ing. Peter Fecilák, PhD.**

Školiace pracovisko: **Katedra počítačov a informatiky**

Konzultant práce: **Ing. Erika Abigail Katonová**

Pracovisko konzultanta: **Katedra počítačov a informatiky**

Pokyny na vypracovanie diplomovej práce:

1. Analyzovať možnosti strojového učenia pre hľadanie ciest v grafových štruktúrach.
2. Navrhnuť model umožňujúci hľadať cesty v grafových štruktúrach pre optimalizáciu smerovacieho procesu.
3. Vytvoriť model pre hľadanie ciest v grafových štruktúrach.
4. Na základe komparatívnej analýzy, vytvoreného modelu so štandardnými smerovacími procesmi a algoritmami, vyhodnotiť efektivitu vytvoreného modelu.
5. Dokumentáciu spracovať podľa katedrových štandardov.

Jazyk, v ktorom sa práca vypracuje: **slovenský**

Termín pre odovzdanie práce: **24.04.2026**

Dátum zadania diplomovej práce: **31.10.2025**



n. z. Prof. Ing. Liberios Vokorokos
.....
prof. Ing. Liberios Vokorokos, PhD.

dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval samostatne s použitím uvedenej odbornej literatúry, ako aj s využitím nástrojov generatívnej umelej inteligencie v súlade s Metodickým pokynom o záverečných a kvalifikačných prácach a Etickým kódexom študenta Technickej univerzity v Košiciach. Umelá inteligencia bola použitá ako podporný nástroj bez porušenia zásad akademickej etiky a autorskej integrity.

Podakovanie

Na tomto mieste by som rád poďakoval svojmu vedúcemu práce a svojej konzultantke za ich čas a odborné vedenie počas riešenia mojej záverečnej práce. Rovnako by som sa rád poďakoval svojim rodičom a priateľom za ich podporu a povzbudzovanie počas celého môjho štúdia.

Obsah

1	Úvod	1
2	Úvod do problematiky smerovania	4
2.1	Softvérovo definované siete (SDN)	4
2.2	Zámerovo orientované siete (IBN)	5
3	Možnosti smerovania v SDN	8
3.1	Hlboké učenie posilňovaním	9
3.2	Využitie učenia s učiteľom	14
3.3	Predikcia stavu siete	16
4	Sumarizácia analyzovaných prístupov	19
4.1	Možnosti realizácie smerovania	19
4.2	Predikcia vlastností komunikácie a siete	20
4.3	Používané prístupy strojového učenia	21
4.4	Trénovanie modelov a agentov	22
4.5	Realizácia porovnaní výsledkov	22
5	Návrh riešenia	24
5.1	Všeobecný návrh systému	24
5.1.1	Fyzická sieť	24
5.1.2	Centrálny kontrolér	25
5.2	Smerovanie s využitím strojového učenia	28
5.3	Navrhované prístupy pre smerovanie s využitím strojového učenia	29
5.3.1	Predikcia metriky hrán	30
5.3.2	Výber cesty	33
5.3.3	Kombinovaný prístup	34
5.4	Metódy strojového učenia	35
5.4.1	Algoritmus TD3	36
5.4.2	Algoritmus PPO	39

5.4.3	Algoritmus DQN	40
5.5	Prostredie	43
5.5.1	Modelované prostredie	44
5.5.2	Model siete	45
5.5.3	Akcie agenta	48
5.5.4	Reprezentácia stavu	49
5.5.5	Reprezentácia odmeny	51
5.5.6	Experimentálne topológie	53
6	Opis implementácie	56
6.1	Implementácia prostredia	56
6.1.1	Prostredie	57
6.1.2	Reprezentácie stavu	58
6.1.3	Reprezentácie odmeny	59
6.1.4	Topológie	59
6.2	Implementácia sieťového modelu	63
6.3	Implementácia smerovacích prístupov	64
6.3.1	Agent pre predikciu metriky	65
6.3.2	Agent pre výber cesty	66
6.3.3	Kombinácia oboch prístupov	68
7	Priebeh experimentov	69
7.1	Experimenty	69
7.2	Trénovanie	72
8	Vyhodnotenie	78
8.1	Dosiahnuté výsledky samostatných prístupov	79
8.2	Dosiahnuté výsledky najlepších prístupov	88
8.3	Zhrnutie	94
9	Záver	96
	Zoznam použitej literatúry	100
	Zoznam skratiek	106
	Zoznam príloh	108
A	Používateľská príručka	109

Zoznam obrázkov

2.1	Znázornenie aplikácie IBN ¹	6
5.1	Diagram navrhovaného systému pre smerovanie dátových tokov .	25
5.2	Topológie využité v experimentoch	54
7.1	Výsledky tréovania dosiahnuté agentom pre samostatnú predikciu metriky počas tréovania	74
7.2	Výsledky tréovania dosiahnuté agentom pre súčasnú predikciu metriky počas tréovania	75
7.3	Dosiahnuté výsledky pri tréovaní agenta pre výber cesty s využitím DQN	76
7.4	Dosiahnuté výsledky pri tréovaní agenta pre výber cesty s využitím PPO	77
8.1	Strata paketov dosiahnutá pri experimentoch pri využití rôznych prístupov k predikcii váh liniek.	80
8.2	Oneskorenie v milisekundách dosiahnuté pri experimentoch pri využití rôznych prístupov k predikcii váh liniek.	81
8.3	Priemerné počty prázdnych liniek pri experimentoch pri využití rôznych prístupov k predikcii váh liniek.	82
8.4	Priemerné počty tokov dát na linkách pri experimentoch pri využití rôznych prístupov k predikcii váh liniek.	83
8.5	Strata paketov dosiahnutá pri experimentoch pri využití rôznych prístupov k výberu cesty.	84
8.6	Oneskorenie v milisekundách dosiahnuté pri experimentoch pri využití rôznych prístupov k výberu cesty.	85
8.7	Priemerné počty prázdnych liniek pri experimentoch pri využití rôznych prístupov k výberu cesty.	86
8.8	Priemerné počty tokov dát na linkách pri experimentoch pri využití rôznych prístupov k výberu cesty.	87

8.9	Strata paketov dosiahnutá pri experimentoch pri využití rôznych prístupov k smerovaniu.	89
8.10	Oneskorenie v milisekundách dosiahnuté pri experimentoch pri využití rôznych prístupov k smerovaniu.	90
8.11	Strata paketov dosiahnutá pri experimentoch pri využití rôznych prístupov k smerovaniu, separátne dáta pre vybrané topológie. . .	91
8.12	Priemerné počty prázdnych liniek pri experimentoch pri využití rôznych prístupov k smerovaniu.	92
8.13	Priemerné počty tokov dát na linkách pri experimentoch pri využití rôznych prístupov k smerovaniu.	93

Zoznam tabuliek

5.1	Sumarizácia navrhovaných topológií	54
7.1	Sumár vlastností a nastavení topológií	70
7.2	Hyperparametre použité pre tréovanie agenta pre predikciu met- riky	72
7.3	Parametre Ornstein-Uhlenbeck šumu	73
7.4	Hyperparametre použité pri tréovaní agenta pre výber cesty s vy- užitím DQN	76
7.5	Hyperparametre použité pri tréovaní agent pre výber cesty s vy- užitím algoritmu PPO	76
8.1	Priemerná strata paketov dosiahnutá pri experimentoch smerova- cím prístupmi podľa topológie.	88
8.2	Priemerné oneskorenie v milisekundách dosiahnuté pri experimen- toch smerovacím prístupmi podľa topológie.	91
8.3	Priemerný počet dátových tokov na linke dosiahnutých pri expe- rimentoch smerovacím prístupmi podľa topológie.	94
8.4	Priemerný počet prázdnych liniek dosiahnutých pri experimen- toch smerovacím prístupmi podľa topológie.	94

Zoznam výpisov

6.1	Dátová štruktúra topológie	59
6.2	Dátová štruktúra uzla	60
6.3	Dátová štruktúra rozhrania	61
6.4	Dátová štruktúra hrany	62

1 Úvod

Jedným z najčastejších odvetví vývoja v dnešnej dobe je automatizácia procesov, s ktorými sa stretávame v každodennom živote. Okrem automatizácie nových procesov, sa často stretávame aj s možnosťou vylepšenia automatizácie už plne automatizovaných procesov. Prácu plne automatizovaných procesov by mohlo byť možné zefektívniť vylepšením už existujúcich, a nimi využívaných, automatizačných prístupov, alebo využitím nových poznatkov aj z iných odvetví pre zlepšenie existujúcich procesov. Jedným takým prípadom, kde by bolo možné zlepšiť efektívnosť automatizovaných procesov, je aj smerovanie v počítačových sieťach.

Ako je očividné, smerovanie v počítačových sieťach je plne automatizované. Ľudia môžu do procesu smerovania zasahovať primárne len úpravami v pravidlách pre smerovanie dátových tokov. Ak je počítačová sieť necentralizovaná, každé sieťové zariadenie rieši samostatne problém toho, kam má poslať komunikáciu, ktorá na zariadení vznikne, respektíve, ktorú prijme od iného zariadenia, ale ktorej cieľom nie je samotné zariadenie. Pre riešenie problému komunikácie medzi zariadeniami v počítačových sieťach existuje veľké množstvo protokolov, ktoré zabezpečujú správnosť komunikácie medzi všetkými zariadeniami prepojenými v spoločnej sieti.

Centralizované riadenie počítačovej siete dovoľuje preniesť zodpovednosť za smerovacie rozhodnutia z jednotlivých sieťových zariadení na centralizovaný bod, ktorý zabezpečí smerovacie rozhodnutia pre celú počítačovú sieť. Takéto riešenie môže mať výhody oproti necentralizovaným riešeniam v tom, že centralizovaný kontrolér môže robiť smerovacie rozhodnutia pri jednotlivých požiadavkách pre vytvorenie komunikačnej cesty medzi dvoma bodmi v sieti. Tým vieme potenciálne lepšie využiť redundantné cesty medzi zariadeniami, ktoré by pri necentralizovaných riešeniach mohli ostať čiastočne, v niektorých situáciach až takmer úplne, nevyužívané. Takýto systém taktiež môže rýchlejšie a cielenejšie reagovať na výpadok linky alebo sieťového zariadenia. Centralizované riadenie má, samozrejme, aj svoje nevýhody, ako je dlhšie oneskorenie pri počiatkovej komunikácii spojené s požiadavkou o nájdenie smerovacej cesty medzi dvoma zariadeniami.

Takéto zdržanie je spojené nielen s logikou spojenou so smerovaním, ale aj s nutnosťou posielania požiadavky na centralizovaný kontrolér cez sieť a s prenosom nových smerovacích informácií na všetky zariadenia v sieti. Ďalším problémom, potenciálne serióznejším, je len jeden bod zlyhania pre smerovanie celej počítačovej siete. Tento problém sa dá čiastočne zmierňovať protokolmi, ktoré by zabezpečovali prechod z centralizovaného riadenia počítačovej siete na necentralizovanú, v prípade zlyhania komunikácie s centralizovaným kontrolérom, alebo aj existenciou záložného kontroléra, ktorý prevezme funkciu primárneho kontroléra pri jeho zlyhaní alebo nedosiahnuteľnosti.

Tu potenciálne vyvstáva otázka, akým spôsobom máme nachádzať cesty medzi dvoma zariadeniami. Jednou možnosťou je využívať existujúce algoritmy pre smerovanie dát v počítačovej sieti. V tomto prípade môžeme vychádzať z bežne používaných algoritmov pri smerovaní, ktoré sa v dnešnej dobe využívajú, a len ich modifikovať pre naše mierne odlišné potreby. Keďže, v tomto prípade, by sme chceli potenciálne využívať možnosť vytvárania individuálnych ciest pre smerovanie komunikácie medzi dvoma koncovými bodmi, vedeli by sme pri každom takomto dopyte o novú cestu hľadať čo najlepšiu, v ideálnom prípade optimálnu, cestu medzi týmito koncovými bodmi, pričom by sme vedeli zohľadňovať aktuálny stav počítačovej siete pri týchto rozhodnutiach. Práve tu je potenciálna možnosť využívať strojové učenie pre smerovanie tokov dát v počítačových sieťach.

Strojové učenie je jeden z často využívaných prístupov pre automatizáciu procesov. V počítačových sieťach je možné strojové učenie využívať vo viacerých prípadoch, napríklad

- pri sledovaní trendov komunikácie v počítačovej sieti,
- pri odhaľovaní podozrivej komunikácie, útokov,
- pri samotných smerovacích rozhodnutiach.

Jednou z výhod strojového učenia pri smerovaní by mohlo byť to, že pre nájdenie optimálnych smerovacích rozhodnutí sú často potrebné komplexné algoritmy. Čo môže byť, samozrejme, značný problém pri smerovaní, keďže akékoľvek oneskorenie je nežiadúce pri smerovaní dát. V tomto prípade by prístupy strojového učenia práve mohli znížiť časovú a priestorovú komplexitu nájdenia dobrého smerovacieho rozhodnutia pre tok dát medzi dvoma koncovými bodmi. Táto práca je práve venovaná vhodným prístupom strojového učenia pre smerovanie tokov dát v počítačových sieťach.

V tejto práci bude kladená pozornosť hlavne na dva prístupy k využitiu strojového učenia pri smerovaní a ich kombinácii do jedného prístupu:

- výber cesty z množiny najkratších ciest,
- predikcia metriky na základe stavu siete alebo linky,
- a spojením oboch týchto prístupov.

Tieto prístupy budú v tejto práci porovnané medzi sebou, ako aj s výberom cesty na základe algoritmu pre hľadanie najkratších ciest v grafe. Budú využité dve metriky pre algoritmus hľadania najkratších ciest v grafe:

- metrika sledujúca počet skokov potrebných pre komunikáciu medzi dvoma zariadeniami, podobne ako protokol RIP,
- a metrika sledujúca relatívny pomer referenčnej rýchlosti linky predelenú aktuálnou rýchlosťou linky, podobne ako protokol OSPF.

Pri porovnávaní bude hlavne sledovaná stratovosť a oneskorenie komunikácie v počítačovej sieti pri využití všetkých spomínaných prístupov pri tvorbe smerovacích rozhodnutí.

Formulácia úlohy

Úlohou tejto práce je analyzovať možnosti využitia strojového učenia pri smerovaní tokov dát v centrálne riadených sieťach. Za týmto účelom je potrebné identifikovať existujúce návrhy metód pre smerovanie komunikácie v počítačových sieťach s využitím strojového učenia. K už existujúcim prístupom je možné pridať nové navrhované prístupy, ktoré môžu, ale nemusia, vychádzať z už existujúcich metód. Taktiež je potrebné porovnať jednotlivé nájdené metódy a analyzovať ich vhodnosť využitia v centrálne riadenej počítačovej sieti. Následne je možné tieto prístupy implementovať a experimentálne porovnať výsledky dosiahnuté jednotlivými prístupmi na modeli počítačovej siete. Porovnanie je taktiež možné vykonať s algoritmickým prístupom založeným na nájdení najkratšej cesty v grafe (topológii siete) na základe metriky odvodennej od počtu skokov medzi zdrojom a cieľom komunikácie a metriky založenej na pomere referenčnej hodnoty šírky pásma a šírky pásma linky. Na základe týchto porovnaní bude možné určiť najvhodnejšie prístupy pre smerovanie komunikácie v počítačovej sieti, respektíve, určiť, v ktorých prípadoch sú jednotlivé identifikované prístupy vhodné.

2 Úvod do problematiky smerovania

V dnešnej dobe sa často v spojení s počítačovými sieťami diskutuje o možnosti centralizovaného manažovania sietí. Keďže mnohé práce využívajúce strojové učenie pri práci s počítačovými sieťami, opísané v kapitole č. 3, využívajú práve prístupy centrálneho riadenia siete, je potrebné spomenúť a krátko opísať tieto prístupy. V tejto kapitole budú v krátkosti spomenuté Softvérovo definované siete (anglicky Software-Defined Networking) a Zámerovo orientované siete (anglicky Intent-Based Networking).

2.1 Softvérovo definované siete (SDN)

Softvérovo definované siete SDN [1, 2] je názov pre prístup riadenie sietí, ktorý oddeľuje riadiacu rovinu od dátovej roviny takým spôsobom, aby sa smerovacie rozhodnutia vykonávali na centrálnom zariadení. To umožňuje ovplyvňovať sieť a jej správanie dynamicky za pomoci softvéru nezávisle na technológiách používaných v sieti.

Výhody a nevýhody SDN

Takýto prístup má viaceré výhody a nevýhody, ktoré boli sumarizované v práci [1] autorov Hakiri et al. Výhodami môže byť jednoduchšie manažovanie siete a lepší prehľad o jej stave, keďže je sieť možné riadiť a monitorovať z centrálneho bodu (kontrolér).

Veľkou nevýhodou je zase problém potenciálneho zlyhania centrálneho bodu, ktorý môže jednoducho zapríčiniť znemožnenie komunikácie v celej sieti. To je práve zapríčinené tým, že ovládač zodpovedá za smerovacie rozhodnutia v celej sieti.

Ďalšou nevýhodou je potreba väčšieho množstva riadiacej a oznamovacej komunikácie medzi centrálnym bodom a zariadeniami prepájajúcimi sieť. To ovplyvňuje šírku voľného pásma pre smerovanú komunikáciu, ako aj nutnosť väčšieho množstva strojového času pre jednotlivé zariadenia pri sledovaní zmien

na rozhraniach a informovaní ovládača.

Taktiež je na základe politík potrebné každý nový tok komunikácie klasifikovať osobitne. V tomto prípade je ale nutné komunikovať s ovládačom, čo môže spôsobovať spomalenie iniciálnej komunikácie a nutnosť častej komunikácie medzi zariadeniami a ovládačom.

Škálovateľnosť takéhoto riešenia je tiež problematická, keďže každé nové rozšírenie siete predstavuje novú výpočtovú záťaž pre ovládač. To môže do značnej miery predĺžiť čakanie na smerovacie rozhodnutia, čím môže nastať spomalenie práce siete alebo až jej výpadky z pohľadu používateľov.

2.2 Zámerovo orientované siete (IBN)

Zámerovo orientovaná sieť IBN je nový prístup k spravovaniu a návrhu sietí, ktorý je, podľa niektorých autorov [3], založený na vybraných fundamentálnych princípoch SDN. Myšlienka tohto prístupu [4, 5] je ale založená na abstraktnom definovaní siete pomocou zámeru (anglicky intent). To by malo dovoliť jednoduchšiu správu a konfiguráciu siete, ako aj umožňovať lepšie zachytávať potreby používateľov.

Zámer v tomto kontexte môže byť definovaný [6] ako „abstrakcia, vysoko úrovňová abstrakcia využitá na operáciu siete“. Z toho podľa autorov [4] vyplýva, že cieľom systému založenom na IBN je umožniť definovať správanie a konfiguráciu siete pomocou nariadení, pri ktorých nie je potrebné špecifikovať presný postup krokov alebo konfiguráciu. Systém by sám mal vedieť dokázať zmeniť správanie podľa potrieb zákazníkov.

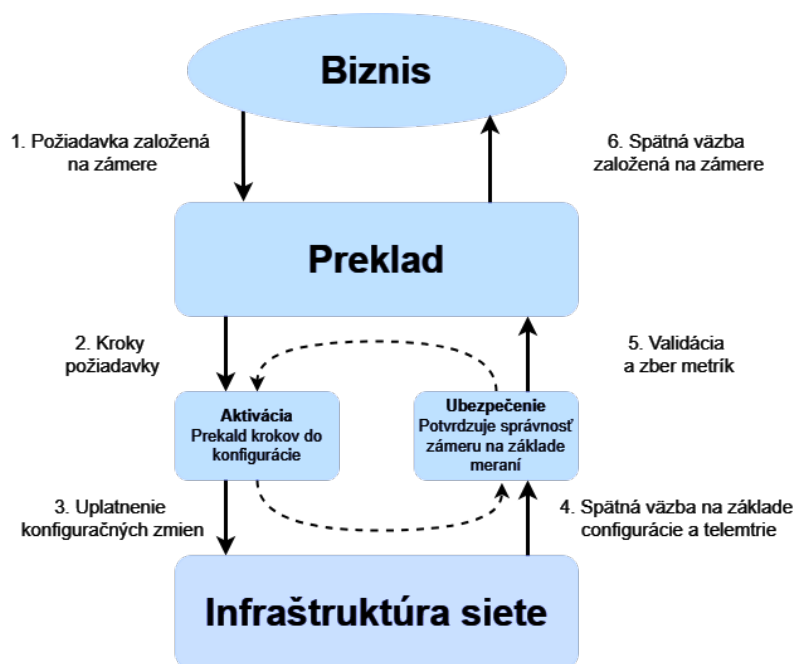
Pre potreby splnenia podmienok kladených na systém [4] založený na myšlienke IBN, systém musí implementovať viaceré funkcie. V nasledujúcom paragrafe spomenieme niektoré z nich.

Systém musí dokázať spracovať zámer od užívateľa a interpretovať ho tak, aby bolo možné aplikovať zistenia do konfigurácie nastavení siete. Systém taktiež musí vedieť monitorovať stav siete a samostatne vyhodnocovať stav siete na základe získaných dát. Na základe aktuálneho stavu siete by mal vedieť dynamicky robiť úpravy pre zachovanie platnosti zámeru pri zmenách správania siete. V neposlednom rade je potrebné umožniť kontroly stavu siete užívateľovi systému, ako aj vytvoriť možnosť pre uistenie užívateľa o správnosti prevádzky siete.

Aplikácia IBN

Z pohľadu konkrétnej aplikácie tohto prístupu v praxi, napríklad podľa spoločnosti Cisco², je možné používať účel (anglicky purpose) ako synonymum k zámeru. To znamená, že cieľom používateľa je definovať účel a zmysel bez toho, aby mal presne definovať kroky potrebné pre naplnenie stanoveného cieľa.

To sa v tomto prípade implementuje ako postupnosť cyklicky opakovaných krokov, ktoré môžeme vidieť na obr. 2.1.



Obrázok 2.1: Znázornenie aplikácie IBN³

Užívateľ, v tomto prípade definovaný ako biznis, zadá požiadavky na základe jeho zámeru, potrieb. Tento zámer je následne preposlaný translačnému procesu, kde je preložený do krokov, ktoré je potrebné aplikovať, aby sa zámer užívateľa (biznisu) uplatnil a prejavil v samotnej sieti. Následne sú tieto kroky predávané do takzvaného aktivačného procesu.

Cieľom aktivačného procesu je transformovať tieto kroky do konfigurácií pre konkrétne zariadenia. Následne musí tento proces dostať konfigurácie do konkrétnych zariadení, to znamená, že vytvorí k nim spojenie a pošle im konfigurácie, ktoré musia nastaviť.

Po aplikácii zmien je potrebné sledovať správanie siete, aby sme mohli získať spätnú väzbu o zmenách vykonaných v sieti. Na to nám môžu slúžiť rôzne metriky a analýzy dátových tokov v sieti. Získané merania sa posielajú do procesu, ktorý zodpovedá za uistenie o správnosti zmien v sieti.

²<https://www.ciscopress.com/articles/article.asp?p=2995353>

Tento proces má za úlohu vyhodnotiť vykonané merania a zistiť, či je splnený zámer, ktorý bol na začiatku zadáný užívateľom. Proces uistenia dáva spätnú väzbu aktivačnému procesu, ktorý môže upraviť aplikované konfigurácie na základe získaných dát z prevádzky siete. Zistenia vo forme metrík, potenciálnych problémov a iných validačných dát sa posielajú ďalej na proces zodpovedný za preklad zámeru.

Proces tieto zozbierané dáta analyzuje oproti pôvodne definovanému zámeru. Pri zistení problémov, má za úlohu informovať o chybnom stave siete užívateľov (správcov), ktorí by tieto problémy mali riešiť, napríklad pomocou zmeny zadaného zámeru do systému. Taktiež slúži na celkový zber dát súvisiacich s fungovaním siete a umožňuje ich zobrazovať pre vyhodnotenie práce siete užívateľom.

Výhoda IBN v takomto prípade je hlavne v možnosti samovalidácie uplatnených zmien v konfiguráciách sieťových zariadení. To umožňuje systému reagovať na nepriaznivé zmeny a automaticky sa pokúsiť napraviť vzniknuté problémy v sieti. Užívateľ by mal byť pri takomto systéme nútený reagovať len pri chybách, z ktorých sa systém sám nedokáže zotaviť alebo v prípade zlej definície zámeru užívateľom.

Samozrejme, aj samotná automatizácia aplikácie konfigurácií jednotlivých sieťových zariadení je veľkou výhodou, ale v tejto implementácii sa dáva väčšia pozornosť automatizovanej validácii konfiguračných zmien zariadení a ich vplyvu na celkové fungovanie siete.

3 Možnosti smerovania v SDN

V každej sieti je potrebné umožniť posielanie komunikácie medzi k nej pripojenými zariadeniami. Jednými zo štandardných smerovacích protokolov, či už dnes menej alebo viac využívaných, sú OSPF [7], IS-IS [8], EIGRP [9], RIP [10] a iné.

V dnešnej dobe sa ale často kladie väčší dôraz nielen na samotné efektívne nájdenie cesty medzi zariadeniami pripojenými k sieti, ale aj na zabezpečenie čo najlepších kvalitatívnych vlastností komunikácie medzi nimi. Taktiež je významná snaha o zlepšenie využívania viacerých ciest medzi komunikujúcimi zariadeniami, hlavne pri vyššej vyťaženosti siete.

Existuje veľa prác zaoberajúcich sa zlepšením smerovania v sieťach práve so zohľadnením kvalitatívnych kritérií komunikácie. Všeobecne tieto práce môžeme rozdeliť na dve skupiny, práce zaoberajúce sa čisto algoritmickým prístupom k smerovaniu komunikácie a práce, ktoré sa snažia využívať strojové učenie pri smerovaní.

Autori Guck et al. [11] spravili prieskum navrhovaných algoritmov pre smerovanie sieťovej komunikácie s ohľadom na požiadavky na kvalitu služieb v rôznych prácach. Autori porovnávali algoritmy na základe ich časovej a cenovej náročnosti, škálovateľnosti, efektivity práce a iného. Podľa autorov, žiaden z nimi porovnávaných algoritmov nie je možné nazvať jednoznačným víťazom. Z ich záverov vyplýva, že v závislosti od požiadaviek a situácie je možné nazvať najlepšou alternatívou rôzne algoritmy.

Prehľad o prácach zaoberajúcich sa možnosťami využitia strojového učenia pri smerovaní komunikácie urobili autori Amin et al. [12]. V svojej práci popisujú rôzne navrhované možnosti využitia strojového učenia pri vykonaní smerovacích rozhodnutí. Oni rozdeľujú práce podľa využitej metódy strojového učenia, to znamená, na práce využívajúce strojové učenie s učiteľom, bez učiteľa a na učenie posilňovaním. Učenie posilňovaním ešte ďalej delia autori na učenie posilňovaním a na hlboké učenie posilňovaním. Autori v svojej práci dospeli k niekoľkým záverom:

- Tieto práce sú často porovnávané len s možnosťami algoritmických imple-

mentácií, najčastejšie s algoritmom Shortest Path First (SPF).

- V prácach sa často nespomínajú všetky podstatné implementačné podrobnosti, ktoré by boli potrebné na replikáciu výsledkov. Z tohto dôvodu sú podľa autorov porovnávané návrhy prác hlavne s SPF algoritmom.
- Pri porovnávaní autori často používajú rôzne topológie, čo taktiež zťažuje porovnávanie navrhovaných prístupov s inými navrhovanými prístupmi.
- Väčšina nimi porovnávaných prác robila porovnanie ich navrhovaného modelu s inými prístupmi na základe latencie a priepustnosti v simulovanom prostredí.

Aj keď táto práca je z roku 2021, ich závery platia aj pre väčšinu prác, ktoré vznikli po ich práci. V nasledujúcich sekciách sú zosumarizované niektoré konkrétne navrhované systémy pre riešenie smerovania v počítačových sieťach.

3.1 Hlboké učenie posilňovaním

Jedným z najčastejšie využívaných prístupov pri smerovaní pomocou strojového učenia pre smerovanie je učenie posilňovaním (anglicky Reinforcement Learning (RL)), v slovenčine učenie posilňovaním. Ak využívame hlboké neurónové siete (anglicky Deep Neural Networks (DNN)) v RL, potom hovoríme o hlbokom učení posilňovaním (anglicky Deep Reinforcement Learning (DRL)). Práve DRL sa najčastejšie využíva v relatívne nových prácach [13, 14, 15, 16, 17] aplikujúcich strojové učenie pri vykonávaní smerovacích rozhodnutí. V tejto časti sú spomenuté niektoré z nich.

Autori Kim et al. [13] v svojej práci používali DRL na sieti riadenej SDN, tento postup bude platiť aj pre ostatné spomenuté návrhy v tejto sekcii, ak nebude explicitne pri nich inak spomenuté. Pritom využívali konkrétne Deep Deterministic Policy Gradient (DDPG) pre vypočítanie váh jednotlivých liniek medzi zariadeniami, na základe ktorých vyhľadávali najlepšiu cestu medzi dvoma zariadeniami v sieti. Trénovanie modelu prebiehalo v offline režime, čo znamená, že model bol trénovaný na simulácii siete, nie pri jeho nasadení do existujúcej topológie. Výhodou takéhoto prístupu je okamžité dobré správanie modelu pri jeho zapojení do smerovania v sieti, ale nevýhodou je jeho neschopnosť sa prispôbovať zmenám siete, či už topologickým alebo v charaktere posielených správ. Podľa ich porovnaní, takýto model dosahoval lepšie výsledky ako s ním porovnávané prístupy, hlavne sa jednalo o tradičné algoritmy, ako napríklad SPF.

Vo svojej práci [18] autori využívajú Co-embedding Graph Convolution Network (CeGCN) pre predikciu váh jednotlivých liniek. Takto predikované linky sú následne využité Dijkstrovým algoritmom pre výber najkratšej cesty medzi zdrojom a cieľom sieťovej komunikácie. Pred tým, ako sú predikované váhy liniek použité, autori navrhujú dodatočný krok pre overenie akceptovateľnosti takto predikovaných váh liniek. Autori vykonávali evaluáciu svojho návrhu na troch topológiách, GEANT2, Abilene, OS3E, pričom dosiahnuté výsledky porovnávali s zjednodušeným agentom nevyužívajúcim CeGCN a algoritmom pre výber najkratšej cesty medzi dvoma bodmi.

Práca [14] sa zaoberá hlavne zlepšením schopností modelov reagovať na zmeny v topológii, respektíve hlavne na zmeny v charaktere komunikácie medzi uzlami v nej. Jedným z hlavných nedostatkov väčšiny riešení využívajúcich strojové učenie vidia v tom, že dané riešenia sa často nedokážu efektívne prispôbovať zmenám, ktoré prirodzene nastávajú v sieti. Preto ich cieľom je umožniť pokračovanie učenia agenta aj počas jeho prevádzky na konkrétnej topológii. Ich systém funguje v dvoch fázach, najprv nastane rozdelenie uzlov topológie na riadiace uzly, ktoré sa neskôr manipulujú modelom, a na uzly, ktoré nie sú z pohľadu smerovania až tak dôležité, tieto uzly sa nemanipulujú. V druhej fáze sa model najprv trénuje na simulácii topológie, na ktorej má byť nasadený. Potom je samotný model nasadený, kde naďalej pokračuje v tréningu na základe ale už reálnej spätnej väzby získanej z jeho smerovacích rozhodnutí. Práve to by mu malo umožniť sa prispôbovať zmenám v topológii. Podľa výsledkov ich testov, takýto model vie dosahovať lepšie výsledky ako algoritmy OSPF a viaccestné smerovanie s rovnakou metrikou (anglicky Equal-Cost Multi-Pathing (ECMP)), ako aj nimi porovnávané navrhované modely DRL.

Ďalšia skupina autorov [15] sa snažila vytvoriť systém, ktorý by bol všeobecne schopný pracovať po dotrénovaní s väčším množstvom topológií. Keďže väčšina návrhov je tréňovaná špeciálne pre niektorú konkrétnu topológiu, či už navrhnutú autormi návrhu alebo na niektorej verejne dostupnej topológii, tieto návrhy nie sú schopné pracovať efektívne pri ich nasadení na inej topológii. Preto v ich prístupe autori zahrnuli pri tréningu priebežné menenie topológie, čo by malo zaručiť lepšie učenie všeobecných vlastností topológií modelom počas tréningu. V tomto prípade bol pri práci využitý model založený na grafových neurónových sieťach Graph Neural Network (GNN). Autori taktiež poskytujú prístup k repozitáru, kde je ich návrh vzorovo naimplementovaný⁴. Podľa výsledkov autorov, model vykazuje lepšie výsledky ako iné modely v jeho kategórii.

⁴<https://github.com/knowledgedefinednetworking/DRL-GNN>

Autori [19] navrhli systém pre centralizované riadenie smerovacích rozhodnutí s využitím DRL. Navrhovaný systém taktiež obsahuje modul, ktorého cieľom je monitorovať aktuálny stav siete. Na základe monitorovaného stavu siete sa vytvára vhodná reprezentácia stavu siete, ktorá je využitá DRL agentom pre vykonanie smerovacieho rozhodnutia. Samotný stav je tvorený množinou potenciálnych ciest, ktoré je možné využívať pre smerovanie dát medzi dvoma koncovými bodmi v sieti. Tento stav je tvorený n -ticou, ktorá hovorí o stave cesty z pohľadu oneskorenia, stratovosti, počtu zahodených paketov, dostupnej šírke pásma a stavu front. Systém bol experimentálne testovaný v docker prostredí, pričom bol porovnávaný s viacerými inými prístupmi k centralizovanému smerovaniu. Podľa výsledkov autorov je ich systém schopný dosiahnuť lepšie výsledky ako výsledky dosiahnuté porovnávanými prístupmi.

Vo svojej nasledujúcej práci, autori [20] predstavili vylepšený návrh pre centralizované smerovanie. Okrem zlepšenia v samotnom smerovaní v centrálne riadených sieťach, autori taktiež rozšírili ciele ich systému o lepšiu prácu s dátovými tokmi, ktoré majú značne rozdielne nároky na potrebnú šírku pásma. Taktiež do svojho systému zahrnuli aj zohľadnenie vyššej priority pre dátové toky, ktoré si vyžadujú nízky čas odozvy, napríklad IP telefónia. Tento navrhovaný systém bol porovnávaný podobne ako ich predchádzajúci návrh, pričom boli dosiahnuté lepšie výsledky ako výsledky porovnávaných prístupov, podobne ako pri ich predchádzajúcom návrhu.

Podobný cieľ si stanovili autori [16] v svojej práci, ale pri riešení daného problému kládli dôraz skôr na možnosť pokračovania učenia sa modelu na samotnej topológii, miesto dôrazu učenia modelu na viacerých topológiách pred jeho nasadením. Tento návrh využíva taktiež GNN pre model v DRL, ako aj predchádzajúci spomínaný návrh. Autori spomínajú ako hlavnú výhodu ich návrhu to, že je ich model schopný využívať online učenie. To zaručuje jeho schopnosť lepšie prispôbovať sa novým topológiám, zmenám v štruktúre topológie a zmenám v prenášanej sieťovej komunikácii, ako návrhy, ktoré sa učia výlučne pred nasadením do siete na konkrétnej jednej topológii. Nevýhodou ale takéhoto prístupu môže byť dlhšie tréningovanie na simulácii navrhovaného modelu pred nasadením na konkrétnu topológiu oproti návrhom, ktoré sú predtrénované na väčšom množstve topológií. Podľa testov autorov, ich systém dosahoval mierne lepšie výsledky ako iné ním porovnávané systémy.

Iným cieľom sa zaoberali vo svojej práci autori [21], ktorých snahou je navrhnúť efektívny spôsob pre smerovanie dát medzi počítačovými sieťami, t.j. autonómnymi systémami. Ich práca je založená na využití RL spolu s Distributed Led-

ger Technology (DLT), založenej na Directed Acyclic Graph (DAG). V ich návrhu je RL, využitie pre výber interných ciest, nazývaných v praci ako pathlet, medzi vstupným a výstupným bodom vo vnútri konkrétneho autonómneho systému. Navrhovaný prístup dosahoval lepšie výsledky v ich porovnaní ako ostatné porovnávané prístupy.

Autori [22] využívajú vo svojom navrhovanom prístupe RL trochu odlišne ako

predchádzajúce spomínané prístupy. Cieľom autorov je predikovať podiel dátovej komunikácie, ktorá bude využívať rozličné možné cesty medzi dvoma bodmi v počítačovej sieti. Autori využívajú algoritmus pre nájdenie oddelených najkratších ciest v grafe medzi dvoma bodmi. Po ich nájdení, autori využívajú PPO (anglicky Proximal Policy Optimization) a GNN na predikovanie podielu dátovej komunikácie, ktorý bude využívať už konkrétnu nájdenú cestu medzi dvoma bodmi v sieti. Pri porovnaní, autori dospeli k výsledkom, že nimi navrhovaný prístup dosahuje lepšie výsledky oproti porovnanému prístupu.

Ako alternatívu pre práce venované centrálnemu riadeniu smerovania môže byť práca [17], ktorá sa snaží vykonávať smerovacie rozhodnutia pomocou DRL aj na úrovni jednotlivých sieťových zariadení. Pre porovnanie autori využili podobnú, ale samozrejme upravenú, architektúru modelu aj pre smerovanie za pomoci centrálného uzla, ktorý len aplikuje smerovacie politiky jednotlivým zariadeniam. V tomto prípade autori pre model pracujúci s DRL použili grafové konvolučné neurónové siete (anglicky Graph Convolutional Network (GCN)). Podľa výsledkov autorov, nimi navrhovaný systém lepšie pracuje v prípade distribúcie riadenia jednotlivým uzlom, autori taktiež robili porovnania s inými návrhmi v svojej práci.

Autori Liu et al. vo svojej práci [23] navrhli centralizovaný systém využívajúci DRL, ktorý pre každý uzol udržiava oddeleného agenta. Pri požiadavke pre nové smerovacie rozhodnutie, systém vykoná predikciu na každom z týchto agentov, kde sa určí nasledujúci skok pri smerovaní z každého uzla. Na základe týchto predikcií sa potom vytvorí cesta, ktorou budú smerované dáta daného dátového toku. Samotní agenti využívajú vo svojej architektúre GCN a GAN vrstvy a agenti sú trénovaní s využitím algoritmu PPO. Pri evaluácii ich prístupu, autori porovnávali ich navrhnutý systém s rôznymi podobnými systémami primárne na dvoch počítačových sieťach, konkrétne GEANT a Abilene.

Vo svojej práci [24] autori využívajú DRL pre predikciu vývoja siete, vo forme predikcie stavu sieťovej komunikácie, a vytvorenie smerovacieho rozhodnutia na základe Dueling DQN. Samotné smerovacie rozhodnutie sa vykonáva na základe

vybraných ciest, z ktorých agent vyberá, a stavová reprezentácia závisí na predikcii stavu sieťovej komunikácie a možných ciest, z ktorých agent vyberá. Autori vykonali evaluáciu ich prístupu na 14 uzlovej topológii.

Autori vo svojej práci [25] využívajú spoluprácu viacerých agentov v trochu inom kontexte. Vo svojom návrhu autori využívajú sériu viacerých identických agentov, pričom jeden z agentov robí smerovacie rozhodnutia pre jeden nový dátový tok. Ostatní agenti na základe vývoja stavu siete postupne posudzujú už existujúce a smerované dátové toky v počítačovej sieti za účelom zlepšenia kvalitatívnych vlastností komunikácie v sieti. V tejto práci všetci agenti vyberajú taktiež z množiny možných ciest medzi cieľom a zdrojom komunikácie. Pre evaluáciu dosiahnutých výsledkov autori využívajú topológiu GEANT2, pričom porovnávajú výsledky dosiahnuté nimi navrhovaným prístupom, samostatne využívaným agentom a tradičnými smerovacími protokolmi OSPF a ECMP.

Práca [26] sa zaoberá smerovaním na iných typoch sietí ako predchádzajúce práce. Autori sa snažia využiť RL pri smerovaní paketov vo viacsokových bezdrôtových sieťach. V tomto prípade sa autori zamerali na iné dôležité aspekty komunikácie, keďže v týchto sieťach je často dôležitejšie udržiavať nízku stratovosť komunikácie medzi uzlami a rýchlo reagovať na zmeny v prítomnosti zdrojov rušenia medzi nimi. Pre druhú vrstvu sieťového modelu komunikácie zvolili autori protokol samoorganizovaná mobilná bezdrôtová sieť (anglicky Mobile Ad hoc Network (MANET)) a vo svojej práci zasahovali výlučne do práce tretej vrstvy. Samotný nimi navrhovaný protokol je reaktívny, to znamená, že reaguje na požiadavky o nájdenie cesty medzi dvoma uzlami siete. Pri takej požiadavke začnú komunikovať medzi sebou susedné uzly, na základe odpovedí jednotlivých susedov sa zvolí najlepší ďalší skok pri smerovaní tohto toku komunikácie. V tomto prípade musí byť na každom uzle agent, ktorý zabezpečuje voľbu ďalšieho skoku pri smerovaní. Podľa autorov, ich prístup dosahuje lepšie výsledky ako bežne používané smerovacie protokoly v takýchto sieťach.

Ďalšia práca, ktorá sa zaoberá smerovaním, ale v inom kontexte je práca [27] autorov Martin et al., kde bolo využité RL pre centralizované smerovanie a určovanie vlnovej dĺžky pre jednotlivé toky komunikácie. Takýto problém je rozdielny od predchádzajúcich smerovacích problémov hlavne v tom, že je potrebné nielen určiť ako sa bude smerovať tok komunikácie z počiatočného uzla do koncového, ale aj akú vlnovú dĺžku má využívať daná komunikácia. Tieto problémy patria do skupiny problémov smerovania a priradovania vlnových dĺžok (anglicky Routing and Wavelength Assignment (RWA)). RWA sa zameriavajú na niektoré dodatočné obmedzenia, ako je maximálny počet možných rôznych

vlnových dĺžok, každý tok musí využívať inú vlnovú dĺžku pri komunikácii, počet vstupných tokov musí byť rovnaký ako počet výstupných tokov. Dodatočne k týmto obmedzeniam sa môžu definovať aj doplňujúce obmedzenia, ako je to, že nie je možné zmeniť vlnovú dĺžku pre tok dát medzi uzlami len na prvom uzle, a zohľadňovať aj maximalizáciu dosahu signálu. Vo svojom riešení využívali autori RL s využitím buď DNN alebo Logistickej regresie. Pokusy robili na dvoch topológiách, jednoduchšej päť uzlovej topológii a komplexnejšej pätnásť uzlovej topológii. Primárne boli jednotlivé pokusy porovnávané medzi sebou, to znamená, porovnávali medzi sebou nimi navrhované prístupy, ale podľa ich meraní času, by pri potrebnej rekonfigurácii topológie malo byť potrebné menej času pri využití RL ako pri využití algoritmov.

3.2 Využitie učenia s učiteľom

V tejto sekcii spomenieme prístupy, ktoré sa snažia smerovať komunikáciu v počítačových sieťach pomocou strojového učenia, ktoré ale nie je možné zaradiť do kategórie RL.

Jednou takouto prácou je práca [28], ktorá sa snaží realizovať smerovanie v Smart Grid sieťach pomocou strojového učenia. Smerovanie v Smart Grid vieme rozdeliť všeobecne do troch častí:

- smerovanie medzi senzormi, respektíve zariadeniami, v jednej domácnosti,
- smerovanie medzi domácnosťami a meracími zariadeniami k centrálnemu zbernému bodu okolia,
- smerovanie medzi zbernými bodmi do riadiaceho centra.

Táto práca sa zaoberá hlavne smerovaním pre druhý spomínaný bod. V takýchto sieťach sa väčšinou využíva protokol Routing Protocol for Low-Power and Lossy Networks (RPL), ktorý ale má niektoré nedostatky, ktoré je potrebné adresovať. Autori sa snažili vylepšiť smerovacie možnosti protokolu RPL pomocou strojového učenia tým, že do využívaného smerovacieho procesu včlenia pri rozhodovaní strojové učenie. V tejto práci na to autori využívajú model CatBoost, jeden z bežne využívaných modelov strojového učenia v dnešnej dobe. Takto upravený protokol RPL bol následne porovnávaný s inými bežnými implementáciami protokolu RPL. Protokoly porovnávali najmä na základe podielu úspešne doručených správ a na priemernom oneskorení smerovaných paketov. Podľa ich

výsledkov protokol využívajúci strojové učenie dosahoval priemerne lepšie výsledky, ale napríklad pri oneskorení často ukazoval vyššie rozdiely oneskorenia doručenia paketov ako iné jeho implementácie.

V inej práci [29] sa autori snažia využiť strojové učenie pre zlepšenie smerovania komunikácie medzi autonómnymi systémami, pričom smerovanie je riadené centralizovane cez SDN. Jedným z problémov využitia centralizovaného kontroléra je ale možné zdieľanie privátnych informácií o jednotlivých autonómnych systémoch. To, samozrejme, nie je žiadúce z pohľadu administrátorov týchto autonómnych systémov. Preto boli vyvinuté protokoly, ktoré vedia najsť najkratšie cesty medzi autonómnymi systémami bez potreby zdieľania privátnych informácií o jednotlivých autonómnych systémoch. Tieto protokoly sú ale často dostatočne výpočtovo náročné. V tejto práci sa snažia autori využiť strojové učenie pre zredukovanie celkovej veľkosti grafu topológie na základe toho, ktoré uzly sú nepravdepodobné v tom, že budú súčasťou niektorej najkratšej cesty v grafe. Následne je docielená menšia výpočtová náročnosť na základe menšieho grafu bez redukovaných uzlov pri hľadaní najkratšej cesty. Pri porovnávaní ich prístupu s pôvodným prístupom, autori dospeli k záveru, že pri redukcii grafu je možné znížiť výpočtové nároky až o 75 percent a komunikačné nároky až o 85 percent. Keďže ale sú niektoré uzly grafu vynechané, dochádza aj k zvýšeniu niektorých metrík ciest. Podľa ich výsledkov sa metriky zvýšili maximálne o 16 percent.

Ďalšou prácou v tejto kategórii je práca [30] autorov Cicioğlu et al., ktorá je venovaná vylepšeniu smerovania v bezdrôtových sieťach riadených za pomoci centralizovaného kontroléra. V ich práci slúži strojové učenie na predikciu ceny liniek medzi jednotlivými uzlami topológie. Keďže sa jedná o bezdrôtovú topológiu, tak dôležité údaje pri odhade ceny liniek sú trochu odlišné od tých, ktoré sa väčšinou využívajú pri práci na iných sieťach. Konkrétne, autori zbierali štandardné údaje o latencii a šírke pásma, ktoré ale rozšírili o dáta o vzdialenosti medzi uzlami a pomeru signálu k okolitému šumu. Po predikcii ceny sú vypočítané hodnoty použité ako vstup pre Dijkstrov algoritmus pre nájdenie najkratšej cesty v grafe. Taktiež autori považujú za dôležité spomenúť, že pre tento problém neexistuje dostupná dátová sada, na ktorej by bolo možné riešenia testovať. Preto bol nimi vytvorený vlastný dataset špeciálne vytvorený pre riešenie predikcie cien liniek v bezdrôtových sieťach. Autor robili porovnanie medzi ich vypočítanými cenami ciest a štandardnými metrikami využívanými v smerovacích algoritmoch. Podľa ich výsledkov, dosahoval ich návrh lepšie výsledky.

3.3 Predikcia stavu siete

Alternatívou využitia strojového učenia v počítačových sieťach k smerovaniu komunikácií, je predpoveď stavu siete. Tu môžeme vidieť hlavne dve možnosti implementácie, ktoré sa v dnešnej dobe používajú:

- Predikcia budúceho vývoja stavu siete [31, 32, 33]. V tomto prípade je cieľom odhadnúť, či pri terajšom vývoji stavu môže v blízkej budúcnosti dôjsť k zhoršeniu kvality poskytovaných služieb.
- Druhá možnosť [34, 35, 36] je predikcia vlastností komunikácie, ako napríklad oneskorenie, na základe aktuálneho stavu siete.

V oboch prípadoch sa snažíme čo najpresnejšie odhadnúť stav siete, síce z iného pohľadu. Cieľom týchto prístupov je hlavne zabezpečiť čo najlepšie a najvyššie údaje o sieti pre optimalizačný algoritmus. Príklady oboch kategórií sú opísané v nasledujúcich sekciách. V poslednej sekcii sú ešte opísané niektoré prístupy pre odhad stavu siete, ktoré ale nie je úplne možné zaradiť do jednej z dvoch predchádzajúcich kategórií.

Predikcia vývoja stavu siete

Nasledujúce tri práce, ktoré sú spomenuté v tejto sekcii, sa zaoberajú predikciou budúceho vývoja siete na základe pravdepodobného vývoja komunikačných tokov v sieti.

Práca [32] autorov Bi et al. sa zaoberá hlavne možnosťou predikcie objemu sieťovej komunikácie mierené na jednu aplikáciu. V ich práci sa evaluácia vykonávala na dátach o množstve dopytov na stránku španielskej wikipédie. Takúto implementáciu by ale bolo možné použiť aj na akúkoľvek inú aplikáciu ako aj na vyťaženie jednej linky medzi zariadeniami v sieti. Na predikciu využívajú tri fázy: očistenie dát, hľadanie krátkodobých závislostí pomocou Temporal Convolutional Network (TCN) a následné hľadanie dlhodobých závislostí pomocou Long Short-Term Memory (LSTM). Svoj model porovnávali hlavne so staršími návrhmi pre predikciu záťaže liniek v počítačových sieťach. Tieto modely boli založené hlavne na modeloch založených na Support Vector Machine (SVM). Podľa výsledkov autorov, ich model dosahoval do značnej miery lepšie výsledky ako iné, nimi porovnávané, metódy.

Autori Xie et al. v svojej práci [33] navrhli model, ktorý by bol schopný predvídať záťaž liniek v počítačovej sieti na základe nielen časových závislostí na lokálnej linke, ale aj na predpokladanom vývoji na okolitých linkách. Taktiež pracujú

s troma typmi časových dát, ktoré spracovávajú pomocou oddelených modulov modelu. Jedná sa o:

- dáta s krátkodobou závislosťou, to znamená vývoj stavu siete v posledných časových krokoch,
- dáta s strednodobou závislosťou, sú to historické dáta, ktoré hovoria o vývoji zaťaženia siete v rámci dňa,
- a dáta s dlhodobou závislosťou, sú to historické dáta, ktoré opisujú vývoj siete v rámci týždňa.

Všetky tieto typy dát sú spracované oddelenou LSTM sieťou, následne sú výsledky zjednotené do jednotného výstupu. Tento výstup je potom spracovaný GNN, ktorý robí aj samotnú finálnu predikciu. Podľa výsledkov autorov je možné povedať, že ich model dosahoval lepšie výsledky ako metódy, s ktorými ho porovnávali. Porovnania robili so siedmimi metódami z okolia roku 2020.

Nasledujúca práca [31] sa snaží predikovať vývoj množstva sieťovej komunikácie na bezdrôtových sieťach. V modeli sa snažia zachytiť priestorové aj časové závislosti v sieti pre čo najpresnejšiu predikciu množstva komunikácie v nasledujúcom časovom kroku. Pre získanie informácie o priestorových vlastnostiach využívajú Convolutional Neural Network (CNN). Výstup CNN ďalej využívajú pre vyhľadávanie časových vlastností siete. V tomto kroku využívajú sieť založenú na vrstvách LSTM. Nakoniec je výstup spracovaný pomocou vrstvy založenej na princípe pozornosti, kde získavajú finálne predikcie o budúcom stave siete. Výsledky porovnávali s inými navrhovanými metódami pre predikciu vývoja záťaže v sieti. Podľa nimi poskytnutých výsledkov, ich model dosahoval lepšie výsledky ako ostatné metódy.

Predikcia vlastností komunikácie

V tejto časti spomenieme tri práce, zaoberajúce sa predikciou niektorých vlastností počítačových sietí na základe aktuálneho stavu siete.

Autori [35] Rusek et al. vo svojej práci navrhujú používať model strojového učenia založený na GNN pre rýchlu a presnú predikciu vlastností siete. Z ich pohľadu hlavnou výhodou použitia strojového učenia je jeho nižšia záťaž na systém oproti matematickým modelom siete a taktiež potenciálne lepšia presnosť predikovaných vlastností siete. Výsledný model bol porovnávaný s modelovaným systémom na základe teórie hromadnej obsluhy. Podľa výsledkov autorov, by

ich model mal dosahovať výsledky približne o jednu tretinu lepšie ako model, s ktorým ho porovnávali.

Práca [34] autorov Huo et al. sa snaží priniesť alternatívu k spomínanej práci autorov Rusek et al. Táto práca využíva miesto GNN vo svojom modeli, sieť založenú na Graph Attention Network (GAN), keďže podľa autorov by takýto typ siete mal byť lepšie schopný získavať informácie z grafu reprezentujúceho sieť oproti GNN. Výsledky spomínané v ich práci by podporovali ich predpoklady, keďže pri porovnaní ich modelu s modelom autorov Rusek et al., ich model dosahoval lepšie výsledky pri predpovedi vlastností siete.

Ďalšou prácou nadväzujúcou na prácu Rusek et al., je práca [36] od autorov Ferriol-Galmés et al. Táto práca využíva podobný prístup k riešeniu možnosti predikcie kvalitatívnych vlastností siete na základe modelu využívajúceho GNN. Autori taktiež porovnávali model s výsledkami dosiahnutými pomocou modelovania siete na základe teórie hromadnej obsluhy. Modely boli porovnávané na základe ich presnosti, ako aj podľa rýchlosti a náročnosti výpočtu pre dosiahnutie výsledkov. Aj v tomto prípade podľa testov autorov, daný model dosahuje lepšie výsledky ako modelovanie na základe teórie hromadnej obsluhy.

Iné možnosti predikcie stavu siete

Okrem dvoch vyššie spomínaných skupín prác zaoberajúcich sa predikciou stavu siete, vieme nájsť aj práce, ktoré sa venujú predikcii pravdepodobnosti úspešného doručenia paketov medzi uzlami siete.

Jednou takou prácou je práca [37], ktorá sa snaží predvídať pravdepodobnosť zahodenia paketu pri jeho smerovaní na nasledujúci skok. Cieľom práce je sa pokúsiť zvýšiť pravdepodobnosť doručenia správy tým, že ak sa nepodarí doručiť správu, tak pri jej znovu posielaní, bude poslaná cez iné rozhranie, pri ktorom je šanca zahodenia nižšia ako pri aktuálne využívanom rozhraní. Na predikciu autori využívajú strojové učenie založené na DNN, kde sa na základe historických dát snažia docieľiť presnú predikciu zahodenia správ posielaných na ďalší skok pri smerovaní komunikácie. Dáta boli zbierané z jedného pracovného dňa a jedného víkendového dňa. Na týchto dátach bol model natrénovaný a následne boli robené pokusy ohľadom pravdepodobnosti zahodenia správ smerovaných niektorým rozhraním.

4 Sumarizácia analyzovaných prístupov

Práce, spomínané v kapitole č. 3, robia prehľad aktuálneho stavu problematiky realizácie smerovania pre zlepšenie kvality komunikačných služieb v počítačových sieťach. Táto kapitola sa venuje sumarizácii zistení odvodených z týchto prác a možnostiam ich neskoršieho uplatnenia.

4.1 Možnosti realizácie smerovania

V tejto sekcii sú zhrnuté často používané prístupy pre realizáciu smerovania v prácach spomínaných v predchádzajúcej kapitole.

Tieto práce vieme rozdeliť na tie, ktoré sa venujú smerovaniu z centralizovaného bodu (kontroléra), pričom väčšinou využívajú k tomu prístup manažovania opísaný v sekcii č. 2.1, alebo sa venujú smerovaniu na úrovni samostatných zariadení.

Z vyššie opísaných skupín sú oveľa bežnejšie práce venované prvej skupine riešení, aj keď je, samozrejme, možné nájsť aj práce zaoberajúce sa druhou skupinou riešení. To je z veľkej časti zapríčinené tým, že pre efektívne zlepšenie vlastností komunikácie v sieti, ako sú napríklad oneskorenie, strata paketov a vyťaženie liniek, nie je dostatočné väčšinou získať informáciu len o jednom komunikačnom uzle, ale je potrebné si vytvárať prehľad o celkovom stave siete. To by malo umožňovať centrálné manažovanie siete.

Menšia skupina prác sa venuje prístupom, ktoré robia rozhodnutia lokálne na zariadeniach. V tomto prípade je často potrebné dodatočne propagovať potrebnú informáciu medzi zariadeniami, aby dané rozhodnutia neboli vykonávané len nad lokálne dostupnou informáciou.

Taktiež je práce možné deliť na tie, ktoré sa snažia vykonávať smerovacie rozhodnutia na základe nových, často neštandardných metód, ale s využitím overených algoritmov pre výber cesty, a na tie práce, ktoré navrhujú vykonávať výber

cesty neštandardným spôsobom.

Pri tomto delení sú bežnejšie práce, ktoré využívajú overené algoritmy pre výber cesty. V tomto prípade sú autormi navrhované nové metriky, ktoré využívajú spomínané algoritmy pri svojich rozhodnutiach. V analyzovaných prácach sú tieto metriky determinované pomocou strojového učenia, ale tieto metriky by mohli byť vytvárané akoukoľvek inou metódou.

V menšej miere boli analyzované práce využívajúce neštandardné spôsoby výberu cesty. Tieto práce ešte vieme rozdeliť na dve skupiny:

- Práce, ktorých metódy priamo vykonávajú smerovacie rozhodnutia s aktuálnym stavom siete.
- Práce, ktoré kombinujú výpočet metrík s neštandardným výberom cesty.

4.2 Predikcia vlastností komunikácie a siete

Okrem smerovania spomínaného v sekcii č. 4.1, sa niektoré analyzované práce venovali predikciám. Tieto predikcie sa môžu snažiť predikovať stav siete, vlastnosti komunikácie, predvídať zahodenie paketov pri komunikácii a iné.

Predikovať stav siete, hlavne pravdepodobnosť zahltenia siete, sa práce snažia na základe rôznej kombinácie informácií o aktuálnom stave siete, o historickom vývoji siete, napríklad v rámci dňa, v rámci týždňa alebo aj z dlhších časových rámcov.

Druhá skupina prác sa snaží predikovať vlastnosti komunikácie na základe aktuálneho stavu siete. V tomto prípade sa nepozeralo na časový vývoj siete a na potenciálny stav siete o určitý čas, ale snažime sa predikovať očakávané vlastnosti komunikácie medzi zariadeniami, ako sú oneskorenie a strata paketov. Tieto vlastnosti sa dajú vypočítať pomocou matematických modelov, ale výpočty by boli pri ich využití veľmi zdĺhavé, čo spôsobuje, že takéto matematické modely nie sú praktické. Preto sa tieto vlastnosti predikujú často pomocou strojového učenia.

Okrem vyššie spomínaných dvoch veľkých skupín sa niektoré práce venujú aj predikciám, ktoré nie je možné zaradiť do týchto dvoch skupín. Analyzovaná bola jedna takáto práca, ktorá sa snažila predikovať pravdepodobnosť zahodenia paketu smerovaného na niektorú cieľovú adresu na základe historických údajov o zahodení paketov.

Tieto prístupy priamo neovplyvňujú smerovacie rozhodnutia, ale môžu byť potenciálne využívané pre ich zlepšenie. To môže byť docielené napríklad pri ich

zahrnutí do metrík.

4.3 Používané prístupy strojového učenia

Väčšina analyzovaných prác pre účely smerovania, opísaného v sekcii č. 4.1, alebo pre účely predikcie stavu siete, opísaného v sekcii č. 4.2, využíva strojové učenie. V tejto sekcii sú zosumarizované hlavné poznatky ohľadom využitia strojového učenia v spomínaných prácach.

Autori prác prevažne využívali strojové učenie s učiteľom a učenie posilňovaním, pričom v oboch prípadoch sa používajú ich varianty využívajúce hlboké neurónové siete (Deep Neural Networks (DNN)).

Strojové učenie s učiteľom sa v prácach prevažne využíva pri predikcii stavu siete alebo predikcii vlastností komunikácie. Pri odhade stavu siete sa väčšinou využívajú rekurentné neurónové siete, keďže ony sú lepšie schopné zachytávať časové vzťahy v dátach. Pri odhade vlastností komunikácie sú využívané neurónové siete schopné získať informácie z vhodnej reprezentácie siete a následne predikovať vlastnosti komunikačných tokov v danej sieti. V tomto prípade sa častejšie využívajú neurónové siete ako CNN alebo GNN.

Pre odhad metrík a aj pre vykonanie smerovacích rozhodnutí sa častejšie využíva hlboké učenie posilňovaním DRL. V prípade odhadu metrík sa periodicky na základe vstupu agenta, vo väčšine prípadov vhodná reprezentácia celej siete, odhaduje agentom nová metrika, ktorá sa následne používa pri smerovaní. V závislosti od návrhu systému, môžu byť tieto odhadované hodnoty ešte dodatočne verifikované pre zlepšenie celkovej výkonnosti systému. Pri týchto návrhoch sa väčšinou využívajú CNN alebo GNN pre spracovanie aktuálneho stavu siete.

V prípade smerovacích rozhodnutí sa taktiež využíva prevažne hlboké učenie posilňovaním, pri ktorom sa ale miesto predikcie metriky predikuje smerovacia cesta. Pri jednoduchšej implementácii má každý uzol v sieti vlastného agenta, ktorý lokálne vyberá, cez ktoré rozhranie sa má komunikačný tok smerovať priamo na základe informácie poskytnutej o rozhraniach a komunikačnom toku. Samozrejme, títo agenti nemusia byť spustení lokálne na danom uzle, ale môžu byť simulovaní aj na centrálnom kontroléri. Pri komplexnejších návrhoch môže byť metrika predikovaná na základe reprezentácie siete pomocou jednej neurónovej siete a na jej základe môže smerovať komunikačný tok ďalšia neurónová sieť.

4.4 Trénovanie modelov a agentov

Táto sekcia opisuje, ako sú v analyzovaných prácach realizované tréovania modelov a agentov.

Modely sú väčšinou tréované v simulovaných prostrediach, keďže často nie je možné realizovať reálne prepojené siete, ale aj to nemusí byť v niektorých prípadoch vhodné, keďže tréovanie týchto modelov alebo agentov by mohlo spôsobiť premenlivé zhoršenie kvality komunikácie v sieti, na ktorej prebieha ich tréovanie. V závislosti od dĺžky tréovania by dané zhoršenie mohlo pretrvávať relatívne dlhú dobu, čo nie je často prijateľné. Z tohto dôvodu a z dôvodu, že agenti a modely sú často tréované na špecifickú sieť, aj samotní autori návrhov vychádzajú z predpokladu, že pred nasadením ich riešenia, bude potrebné ho tréovať na konkrétnu sieť počítačov.

Niektoré práce sa práve snažia riešiť zovšeobecnenie tréovaného správania, čím by docielili využiteľnosť agentov a modelov na viaceré siete. Aj pri takýchto návrhoch sa často predpokladá dotréovanie agenta alebo modelu pre konkrétnu počítačovú sieť.

Pri predikciách stavu siete alebo vlastností komunikácie je často možné využívať reálne dáta zo sietí a aj ich na nich tréovať, keďže oni priamo väčšinou neovplyvňujú smerovanie komunikačných tokov v sieti. Z tohto dôvodu sú tieto modely menej často tréované na simulovaných prostrediach.

4.5 Realizácia porovnaní výsledkov

Všetky práce realizujú porovnávanie dosiahnutých výsledkov s inými prístupmi. Dosiahnuté výsledky jednotlivých návrhov sa často porovnávajú náročne, keďže autori porovnávajú ich dosiahnuté výsledky na rozličných parametroch, napríklad na oneskorení, strate paketov a podobne. Taktiež nie všetky práce realizujú pokusy na topologicky rovnakých sieťach. Keďže, ako už bolo spomínané v sekcii č. 4.4, sa využívajú väčšinou simulované prostredia, nie je ani úplne vhodné porovnávať výsledky dosiahnuté rôznymi autormi medzi sebou.

Z týchto dôvodov je porovnanie prístupov skôr možné realizovať na prezentovaných výsledkoch autorov. Jednou nevýhodou takéhoto prístupu je nemožnosť porovnávať prístupy, ktoré autori prác sami v svojej práci neporovnávali s ich návrhom. To často prináša limitované možnosti porovnávania prístupov z pohľadu výsledkov, keďže niektoré práce svoje návrhy porovnávali len s limitovaným množstvom existujúcich algoritmov alebo alternatívnych návrhov.

Tiež je často náročné realizovať porovnania na základe prác autorov, keďže nie vždy sú poskytnuté dostatočne podrobné informácie o implementácii ich navrhovaného systému.

5 Návrh riešenia

V tejto kapitole bude opísaný návrh riešenia pre centralizované riadenie smerovania s využitím strojového učenia, taktiež budú opísané aj metódy strojového učenia využívané jednotlivými navrhovanými metódami a prostredie, v ktorom budú vykonané experimenty. V sekcii č. 5.1 je opísaný všeobecný návrh systému, jeho časti a princíp fungovania. V sekcii č. 5.2 je detailnejšie spísaný opis problému a cieľ, ktorý má byť dosiahnutý jednotlivými navrhovanými prístupmi pre využitie strojového učenia pri smerovaní. Tieto prístupy sú opísané v sekcii č. 5.3. Posledná sekcia č. 5.5 je venovaná opisu prostredia navrhnutého pre vykonanie experimentov.

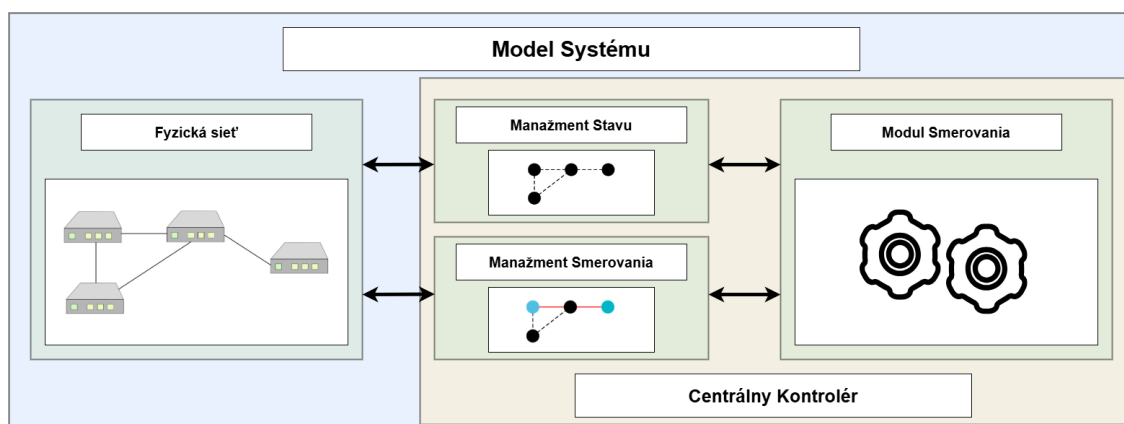
5.1 Všeobecný návrh systému

Táto sekcia je venovaná opisu navrhovaného riešenia zo všeobecného pohľadu nezohľadňujúceho špecifiká konkrétnych navrhovaných prístupov pre smerovanie. Celkovo navrhovaný systém je možné rozdeliť na dve hlavné časti, a to fyzickú sieť a centrálny kontrolér, pričom samotný kontrolér je ďalej zostavený z troch modulov, ktoré sú pomenované Manažment Stavů, Manažment Smerovania a Modul Smerovania, ako je možné vidieť na obrázku č. 5.1. Jednotlivé moduly sú opísané v nasledujúcich podsekcích.

Táto práca je venovaná riešeniu smerovania v počítačovej sieti s využitím strojového učenia, preto bude časť navrhovaného systému opísaná len teoreticky, to jest bez implementácie, a pri vykonaných experimentoch bude len zjednodušene modelovaná.

5.1.1 Fyzická sieť

Fyzická sieť je reprezentovaná reálnymi fyzickými zariadeniami, ktoré tvoria počítačovú sieť, a prepojeniami medzi nimi. Keďže je v tomto návrhu potrebné realizovať centrálné riadenie týchto zariadení, je nevyhnutné, aby všetky zariadenia, ktoré sú využívané pre smerovanie v počítačovej sieti boli schopné meniť svoje



Obrázok 5.1: Diagram navrhovaného systému pre smerovanie dátových tokov

smerovacie politiky na základe zmien vyžadovaných centrálnym kontrolérom. Tieto smerovacie politiky je nevyhnutné aplikovať oddelene pre každý tok dát v počítačovej sieti.

Taktiež je potrebné zabezpečiť počiatočnú konektivitu medzi všetkými smerovacími zariadeniami v počítačovej sieti. Táto konektivita sa taktiež bude využívať pre riadiacu komunikáciu medzi zariadeniami. Základné prepojenie je možné dosiahnuť pomocou štandardných smerovacích protokolov, ako napríklad OSPF [7]. Okrem dosiahnutia počiatočnej komunikácie je takýmto spôsobom tiež možné dosiahnuť udržanie komunikácie pre nové dátové toky aj v prípade zlyhania centrálného kontroléra.

Pri normálnom behu siete a centrálného kontroléra, smerovacie zariadenia pošlú požiadavku na centrálny kontrolér pri výskyte nového dátového toku. Po spracovaní požiadavky centrálnym kontrolérom, centrálny kontrolér zabezpečí distribúciu smerovacej politiky pre daný nový dátový tok každému relevantnému smerovaciemu zariadeniu. Následne jednotlivé pakety daného dátového sú smerované na základe novej politiky.

5.1.2 Centrálny kontrolér

Hlavnou úlohou centrálného kontroléra v navrhovanom systéme je uchovávať reprezentáciu stavu počítačovej siete, aktuálnych politík pre smerovanie dátových tokov a reagovať na požiadavky pre nové smerovacie politiky od smerovacích (manažovaných) zariadení pri výskyte nových dátových tokov. Túto funkcionality rozdelíme do troch modulov, ktoré realizujú jednotlivé spomínané funkcie. Tieto moduly sú opísané v nasledujúcich častiach tejto podsekcie. Taktiež ako výhodu modularity systému sa dá považovať to, že je relatívne jednoduché vymeniť jednu konkrétnu implementáciu modulu, napríklad Modulu Smerovania, za inú,

ktorá vykonáva tú istú funkcionálnu iným spôsobom.

Manažment stavu

Modul s názvom Manažment stavu slúži na zbieranie informácií od manažovaných sieťových zariadení za účelom získania prehľadu o aktuálnom stave počítačovej siete. Táto informácia sa následne uchováva a upravuje na základe zistených zmien pri každom novom dopyte o aktuálnu abstraktnú reprezentáciu stavu zariadení počítačovej siete. Tieto čiastkové reprezentácie sa následne používajú na reprezentáciu stavu siete ako celku.

Získaná reprezentácia aktuálneho stavu siete sa používa Modulom smerovania pre vykonanie smerovacích rozhodnutí. Taktiež keďže táto informácia sa v navrhovanom systéme zbiera počas celého behu systému, je možné zozbieranú informáciu o stave počítačovej siete ďalej využívať pre jej monitorovanie a manažovanie. Takéto využitie zbieraných dát ale nebude realizované v tejto práci, len je tu spomínané ako potenciálne možné rozšírenie navrhovaného systému.

Manažment smerovania

Druhý modul, nazývaný Manažment smerovania, slúži na udržiavanie zoznamu aktívnych dátových tokov a s nimi asociované smerovacie politiky, pričom jeho ďalšou funkciou je reagovať na požiadavky smerovacích zariadení a zabezpečiť distribúciu smerovacích politík pre aktívne dátové toky.

Keď centrálny kontrolér dostane požiadavku pre smerovaciu politiku pre nový dátový tok, najprv modul Manažment smerovania skontroluje, či už neexistuje záznam o danom dátovom toku v tabuľke všetkých dátových tokov. Ak existuje, tak len aplikuje už existujúcu politiku na smerovacie zariadenia. Ak daný záznam neexistuje, tak Manažment smerovanie konzultuje Modul smerovania, pričom očakáva od Modulu smerovania novú politiku pre smerovanie daného dátového toku. Po jej obdržaní zaznamená novú smerovaciu politiku a s jej dátovým tokom do zoznamu všetkých smerovacích politík, a následne aplikuje smerovacie politiky na manažované smerovacie zariadenia.

Modul smerovania

Posledný modul sa stará o vykonanie smerovacích rozhodnutí na požiadavku modulu Manažment smerovania, pričom využíva informáciu o smerovanom toku dát a aktuálnom stave počítačovej siete. V tejto časti je tento modul opísaný všeobecne, pričom konkrétne navrhované implementácie sú opísané v sekcii č.

5.3. Modul smerovania nájde cestu, spájajúcu zdroj a cieľ, získanú na základe poskytnutej informácie o dátovom toku, a na základe aktuálneho stavu manažovanej počítačovej siete, ktorý získa od modulu Manažment stavu. Táto cesta môže byť reprezentovaná množinou uzlov, to jest smerovacích zariadení, alebo hrán, to jest liniek medzi nimi, cez ktoré bude daný dátový tok smerovaný. Táto nájdená cesta je následne poskytnutá modulu Manažment smerovania, ktorý zabezpečí aplikáciu novej smerovacej politiky. Modul smerovania môže byť implementovaný rozličnými spôsobmi, napríklad s využitím algoritmu pre výber najkratšej cesty SPF (anglicky Shortest Path First (SPF)) algoritmu pre nájdenie cesty, pričom metriky, inak povedané váhy liniek, sa môžu meniť v čase, napríklad pomocou strojového učenia, ako bude opísané v sekcii č. 5.3.

Algoritmus 1 Algoritmická reprezentácia smerovania s využitím Modulu smerovania

```

1: procedure ROUTE
2:   state ← create_state(collect_data())
3:   while true do
4:     decrement_ttl_and_check(state)
5:     data ← collect_data()
6:     state ← update_state(state, data)
7:     new_flow ← get_new_flow()
8:     if new_flow then
9:       route ← making_routing_decision(state,
        new_flow)
10:      apply_routing_decision(state, new_flow, route)
11:    end if
12:  end while
13: end procedure

```

Algoritmus č. 1 opisuje selekciu smerovacej politiky pre dátové toky v navrhovanom systéme. Najprv sa vytvorí počiatočná reprezentácia počítačovej siete na základe jej aktuálneho stavu. Celý proces smerovania beží v nekonečnom cykle, opísanom v riadkoch 3-12. Na začiatku každého cyklu je potrebné dekrementovať, znížiť, hodnotu *TTL*. Hodnota *TTL* reprezentuje koľko časových krokov má byť dátový tok aktívny počas experimentu. Taktiež ju skontroluje pre každý dátový tok, pričom, ak daná hodnota nadobudne hodnotu nula, je potrebné dátový tok odstrániť. To je vykonané funkciou *decrement_ttl_and_check* v riadku 4. Riadky 5-11 modeluje požiadavku niektorého manažovaného zariadenia o nové

smerovacie rozhodnutie. V riadku 6 je reprezentácia stavu siete upravená na základe aktuálneho stavu siete. Následne sa vytvorí reprezentácia nového dátového toku na základe požiadavky, ako je znázornené v riadku 7. Pre nový dátový tok sa nájde nová cesta. Táto cesta je aplikovaná na všetky relevantné manažované smerovacie zariadenia. Tento proces je demonštrovaný na riadkoch 8-11.

5.2 Smerovanie s využitím strojového učenia

Táto sekcia je venovaná využitiu strojového učenia pre smerovanie tokov dát v počítačových sieťach. V tejto sekcii bude všeobecne opísané využitie strojového učenia pre smerovania tokov dát, zatiaľ čo sekcia č. 5.3 opíše konkrétne prístupy, ktoré sú navrhnuté pre ich využitie v navrhovanom systéme v sekcii č. 5.1. Keďže samotné smerovanie je súčasťou Modulu smerovania, väčšina vecí, ktoré budú v tejto sekcii spomenuté sa týkajú práve implementácie tohto modulu návrhu. Z dôvodu, že aj smerovanie s využitím strojového učenia si vyžaduje informáciu o aktuálnom stave počítačovej siete a novom dátovom toku, aspoň časť informácie v tejto sekcii je relevantná aj pre ostatné dva moduly centrálného kontroléra opísaného v podsekcii č. 5.1.2.

Cieľom Modulu smerovanie je nájsť najkratšiu cestu p_{ij} medzi dvoma uzlami v_i a v_j . Existuje veľké množstvo spôsobov ako nájsť cestu p_{ij} , ako napríklad typicky používané protokoly OSPF, RIP, ECMP, EIGRP, ako aj napríklad rôzne prístupy opísané v kapitole č. 3. Keďže táto práca je primárne venovaná prístupom využívajúcim strojové učenie pre smerovanie dátových tokov v počítačovej sieti, tak v tejto sekcii bude vytvorený všeobecný prehľad o probléme smerovania prevažne s dôrazom na smerovania s využitím strojového učenia.

Ako cieľ pre smerovanie dát v počítačovej sieti si táto práca kladie zlepšenie výkonu počítačovej siete pri prihlíadaní na komunikáciu medzi dvoma cieľovými bodmi v počítačovej sieti. Táto práca si konkrétne kladie za cieľ znížiť množstvo zahodených paketov pre jednotlivé aktívne dátové toky v počítačovej sieti v každom časovom kroku modelovanej počítačovej siete. Všeobecnejšie by sme vedeli cieľ optimalizácie definovať ako

$$\pi = \arg \min_{\pi} \text{performance_evaluation}(\pi), \quad (5.1)$$

kde π^* je optimálna smerovacia stratégia, funkcia $\arg \min$ vracia takú smerovacia stratégiu π , ktorá dosahuje najlepšie výkonnostné výsledky na základe funkcie $\text{performance_evaluation}$. Funkcia $\text{performance_evaluation}$ môže zohľadňovať viaceré vlastnosti komunikácie v počítačovej sieti, napríklad, množstvo zaho-

dených paketov, oneskorenie, množstvo stále dostupnej šírky pásma, počet nevyužitých liniek a podobne. Čím je stav komunikácií v počítačovej sieti horší, to jest, vysoké množstvo zahodených paketov, veľké množstvo nevyužitých liniek alebo veľké oneskorenie, tým vyššiu hodnotu funkcia vracia. Ako bolo spomenuté vyššie, táto práca berie do úvahy primárne množstvo zahodených paketov pre vyhodnotenie kvality smerovacej stratégie. Funkcia $\arg \min$ vyhodnocuje získané hodnoty zo všetkých vyskúšaných stratégií π , aby našla smerovaciu stratégiu s najlepšimi výsledkami.

Navrhované prístupy využité v tejto práci vychádzajú z Reinforcement Learning (RL), konkrétnejšie z Deep Reinforcement Learning (DRL), tak ako je opísané autormi [38, 39, 40]. Táto práca bude vnímať smerovací proces ako Markov Decision-making Process (MDP). Sada stavov S je považovaná za postačujúcu pre kompletne opísanie všetkých stavov počítačovej siete. Taktiež bude definovaná sada akcií A , sada odmien R a sada pravdepodobností P , pričom bude definovaný prechod medzi stavmi s a s' , kde $s, s' \in S$, keď bola vykonaná akcia a , $a \in A$, v stave s . Takýto prechod nastal s pravdepodobnosťou p , $p \in P$, a bola dosiahnutá odmena r , $r \in R$. Konkrétna akcia, ktorá bola vykonaná v stave s závisí od aktuálne používanej politiky π tak, že $a = \pi(s)$, kde π je aktuálne používaná politika aktuálne používaného smerovacieho prístupu a s je abstraktný opis stavu počítačovej siete v danom časovom kroku.

Celkovým cieľom prístupov, ktoré budú opísané v sekcii č. 5.3, je nájsť takú smerovaciu politiku π' , ktorá by viedla k čo najlepším výsledkom pri zohľadnení funkcie $\text{performance_evaluation}(\pi')$ na smerovacej stratégii π' . Ideálnym cieľom by bolo dosiahnutie stavu, keď smerovacia stratégia π' by dosahovala výsledky optimálnej smerovacej stratégie π^* , keďže vtedy by boli dosiahnuté optimálne výsledky smerovacou stratégiou. Realistickým cieľom je ale dosiahnuť stav, keď π' sa približuje k výsledkom optimálnej smerovacej stratégie π^* v čo najväčšej miere.

5.3 Navrhované prístupy pre smerovanie s využitím strojového učenia

V tejto sekcii sú opísané navrhované prístupy pre využitie strojového učenia pre smerovanie tokov dát v počítačových sieťach. Na tomto mieste sú opísané navrhované prístupy všeobecne, bez konkretizácie špecifických metód strojového učenia, ktoré budú alebo by mohli byť použité pri implementácii týchto prístupov. Využiteľné metódy strojového učenia budú opísané v sekcii č. 5.4.

Ako aj vyplýva z analýzy aktuálneho stavu problematiky v kapitole č. 3, hlavne je možné využiť tri prístupy k smerovaniu tokov dát pri centralizovanom riadení siete:

- predikcia metriky hrán (liniek) v topológii, na základe ktorých sa vyhľadáva najkratšia cesta,
- výber cesty z množiny n najkratších ciest, pričom táto cesta sa bude následne používať na posielanie daného toku dát,
- a predikcia rozdelenia komunikácie medzi viaceré potenciálne cesty.

Prvé dva prístupy budú implementované a experimentálne odskúšané v tejto práci.

K týmto dvom prístupom je ešte pridaný tretí prístup, ktorý sa bude pokúšať spojiť oba predchádzajúce prístupy do jedného celku, to jest, do jedného smerovacieho prístupu. To znamená, že metrika bude vypočítaná jedným agentom, z tejto metriky bude nájdených n ciest, z ktorých následne vyberie druhý agent cestu, ktorá bude využitá pre smerovanie toku dát v počítačovej sieti.

V nasledujúcich podsekcích budú jednotlivé prístupy dôkladnejšie opísané.

5.3.1 Predikcia metriky hrán

Jedným z prístupov k smerovaniu tokov dát v počítačových sieťach s využitím strojového učenia je predikcia metriky. V tomto prípade sa agent podieľa na smerovaní takým spôsobom, že predikuje metriku liniek (hrán) v topológii počítačovej siete. To môže realizovať pre všetky hrany alebo podľa vlastných stanovených pravidiel, len na niektorých dôležitých linkách, respektíve v každom kroku predikovať metriku len časti zo všetkých liniek siete.

Taktiež je možné rozdeliť tento prístup na dve varianty:

- predikcia metriky všetkých liniek súčasne,
- predikcia metriky každej linky samostatne.

Rozdiely v týchto prístupoch sú opísané v nasledujúcich podsekcích.

V oboch prípadoch je nutné po výpočte metriky nájsť cestu medzi dvoma uzlami topológie. Keďže pri takomto systéme sa metrika v čase mení, nie je možné nájsť vhodné cesty skôr, ako príde požiadavka o spojenie medzi dvoma bodmi v topológii. To, samozrejme, môže výrazne zvýšiť výpočtový čas pre nájdenie novej cesty v topológii pri novej požiadavke oproti prístupom, ktoré už majú všetky cesty predvypočítané.

V tejto práci budú bližšie spomenuté oba spomínané varianty tohto prístupu a aj využité pri experimentoch.

Predikcia metriky pre všetky linky súčasne

Pri využití tejto možnosti sú všetky metriky predikovaných liniek získané z jednej predikcie vykonanej agentom na jednom poskytnutom stave. V tomto prípade má agent globálny pohľad na stav siete z pohľadu liniek a topológie.

Keďže v takomto prípade je nevyhnutné dať agentovi informáciu o všetkých linkách súčasne, agent má dobrý prehľad o celkovom stave topológie, čo je jedna možná výhoda tejto varianty. Ďalšou výhodou tejto varianty implementácie je oveľa menší počet samostatných predikcií, ktoré musí agent robiť pri každom novom výpočte metriky pre danú topológiu, keďže nové váhy všetkých liniek sú už po jednej predikcii známe.

Algoritmus 2 Základný algoritmus pre predikciu metriky každej linky súčasne

```

1: procedure MAKE_ROUTING_DECISION(state, new_flow)
2:    $s \leftarrow \text{create\_state}(\text{state})$ 
3:    $\text{metric} \leftarrow \text{get\_metric}(s)$ 
4:    $\text{route} \leftarrow \text{create\_route}(\text{metric}, \text{state}, \text{new\_flow})$ 
5:   return route
6: end procedure

```

Za nevýhodu v tomto prípade sa dá považovať fakt, že nie je možné samostatne predikovať metriky pre jednotlivé linky, čo znemožňuje efektívne využitie tohto prístupu pre parciálne zmeny metrik jednotlivých liniek počítačovej siete. Taktiež môže byť problematickejšie využitie agenta na viacerých topológiách, inak povedané, jeho generické použitie. Keďže všeobecne sa dá predpokladať, že topológie sú tvorené rôznym počtom hrán a uzlov, je potrebné adaptovať vstup aj výstup takéhoto agenta na potenciálne rôzne veľkosti topológií, ak je potrebné ho používať na rôzne veľkých topológiách. Respektíve je nemožné použiť takýto systém efektívne pre všetky možné topológie, stále bude existovať nejaká topológia, ktorú nie je možné takýmto agentom riadiť.

V pseudokóde č. 2 môžeme vidieť realizáciu súčasnej predikcie všetkých liniek v algoritmickej forme. Najprv je potrebné vytvoriť vhodnú reprezentáciu stavu počítačovej siete pre spracovanie agentom pre predikciu metriky, čo je možné vidieť v riadku č. 2. Na základe daného stavu sa predikujú súčasne metriky

každej linky v počítačovej sieti v riadku č. 3. Potom je potrebné na základe hodnôt metriky nájsť najkratšiu cestu v počítačovej sieti, pričom daná nájdená cesta bude využitá pre smerovanie daného toku dát, ako je možné vidieť v riadkoch č. 4 – 5.

Predikcia metriky pre každú linku samostatne

Táto varianta predikuje výhradne metriku jednej linky na základe aktuálneho stavu len danej linky, respektíve danej linky a uzlov na oboch jej stranách. Preto sa aj dá povedať, že takýto výpočet je lokálny z pohľadu linky a topológie, keďže agent má informáciu len ohľadom linky, ktorej metriku predikuje, a potenciálne uzlov, ktoré daná linka prepája.

Výhodou tejto varianty je hlavne ľahká použiteľnosť natrénovaného agenta na akejkolvek topológii počítačovej siete. Preto by táto varianta mohla byť vhodnejšia na všeobecné, generické použitie agenta, keďže daného agenta by nemalo byť vôbec nutné trénovať pre každú topológiu, na ktorej by mal byť neskôr využívaný. Potenciálnou výhodou takéhoto prístupu by mohla byť možnosť predikcie metriky na samotných sieťových zariadeniach, ktoré by vypočítané metriky poskytovali centrálnemu kontroléru pre ich vyhodnotenie. Tým by centrálny kontrolér mohol byť menej zaťažovaný oproti iným prístupom.

Algoritmus 3 Základný algoritmus pre predikciu metriky každej linky samostatne

```

1: procedure MAKE_ROUTING_DECISION(state, new_flow)
2:    $s \leftarrow \text{create\_state}(\text{state})$ 
3:    $\text{metric} \leftarrow \text{empty\_set}$ 
4:   for link in  $s$  do
5:      $\text{link\_weight} \leftarrow \text{get\_metric}(\text{link})$ 
6:      $\text{metric.append}(\text{link\_weight})$ 
7:   end for
8:    $\text{route} \leftarrow \text{create\_route}(\text{metric}, \text{state}, \text{new\_flow})$ 
9:   return route
10: end procedure

```

Nevýhodou ale je hlavne to, že agent stráca akúkoľvek informáciu o celkovom stave siete pri predikovaní metriky liniek. Z tohto dôvodu môže agent využívať pri predikcii metrick liniek len lokálnu informáciu o stave danej linky a nemôže zohľadniť celkový stav siete pri svojom rozhodovaní.

Keďže tento prístup bude využitý pri experimentoch, je vhodné ho opísať aj vo forme pseudokódu. V algoritme č. 3 je opísané fungovanie predikcie metriky v navrhovanom systéme. Najprv je potrebné vytvoriť stav siete, čo je možné vidieť v riadku 2. Následne sa inicializuje prázdna sada metrík a predikujú sa nové metriky pre každú linku v počítačovej sieti, ako je možné vidieť v riadkoch 4 až 7. Potom sa v riadku 8 vyberá najkratšia cesta s využitím novej metriky.

5.3.2 Výber cesty

Druhým prístupom je výber cesty z množiny n najkratších ciest. Tak, ako bolo opísané v analyzovaných prácach v kapitole č. 3, v tomto prípade sa najčastejšie nájdu najkratšie cesty pomocou algoritmu pre hľadanie najkratších ciest v ohodnotenom grafe. Hrany tohto grafu sú ohodnotené nejakou metrikou, napríklad metrikami, ktoré sa využívajú v bežných smerovacích algoritmoch ako je RIP a OSPF. Následne sa k daným cestám vytvorí vhodný stav, ktorý opisuje vlastnosti jednotlivých ciest. Na základe tohto stavu následne agent vyhodnotí, ktorá cesta sa má používať na smerovanie toku dát.

Pri takejto definícii prístupu je ho jednoducho možné použiť na ľubovoľnej topológii počítačovej siete bez ohľadu na jej veľkosť bez akýchkoľvek úprav. Stav bude stále definovaný pomocou pevne daného počtu možných ciest pre výber a vlastnosti každej cesty.

Algoritmus 4 Základný algoritmus pre výber cesty z množiny ciest

```

1: procedure MAKE_ROUTING_DECISION(state, new_flow, n)
2:    $paths \leftarrow \text{select\_n\_paths}(\text{state}, \text{new\_flow}, n)$ 
3:    $s \leftarrow \text{create\_state}(\text{state}, paths)$ 
4:    $route \leftarrow \text{select\_route}(s)$ 
5:   return route
6: end procedure

```

Nevýhodou ale môže byť väčšia náročnosť z pohľadu nájdenia možných ciest na výber pre agenta. Tieto cesty musia byť nájdenie pred rozhodnutím samotného agenta, keďže ony definujú vstupný stav pre agenta. V prípade metrík, ktoré sa v čase nemenia, je možné cesty pripraviť dopredu, a tým znížiť výpočtovú a časovú záťaž na centrálny kontrolér. Samozrejme, ak sa metriky v čase menia, tieto cesty je nutné hľadať v momente, keď je požiadavka na nájdenie cesty.

V algoritme č. 4 je opísané fungovanie výberu cesty z množiny ciest v navrhovanom systéme. Najprv je potrebné vytvoriť množinu n ciest, z ktorých bude

agent vyberať jednu cestu, čo je možné vidieť v riadku 2. Následne sa vytvorí na základe množiny n ciest stav počítačovej siete, ktorý bude reprezentovaný stavmi liniek a uzlov, ktoré sú súčasťou týchto n ciest. To je možné vidieť v riadku 3. V riadku 4 sa následne vyberá najkratšia cesta agent, táto cesta sa bude využívať na smerovanie dát nového dátového toku.

5.3.3 Kombinovaný prístup

Posledným navrhovaným prístupom je spojenie oboch spomínaných prístupov do jedného prístupu. Tento prístup sa dá rozdeliť na dve fázy:

- v prvej fáze je predikovaná metrika a nájdených n najkratších ciest,
- v druhej fáze je z nájdených n ciest vybraná jedna cesta použitá na smerovanie toku dát.

Obe fázy do značnej miery zodpovedajú princípom fungovania samostatných prístupov, na ktorých je kombinovaná metóda založená, a ktoré boli opísané v predchádzajúcich podsekcích. Hlavnou zmenou v prvej fáze oproti samostatnému fungovaniu prístupu pre predikciu metriky je nájdenie n najkratších ciest miesto iba jednej najkratšej cesty. Tieto cesty sú potom vstupom do druhej fázy tohto prístupu miesto statických preddefinovaných n ciest, to jest rozdielom v druhej fáze je len zdroj n najkratších ciest. Táto fáza inak plne zodpovedá prístupu, na ktorom je založená.

Výhodou oproti samostatným prístupom by mohlo byť zmiernenie dopadu možných chýb samostatných prístupov. Taktiež ak jeden prístup má problémy s nejakým faktorom vplývajúcim na správanie sa počítačovej siete, druhý prístup by tento problém mohol byť schopný riešiť.

Nevýhodou môže byť to, že nesprávne fungovanie jednej časti tohto prístupu, môže spôsobiť horšie správanie smerovacieho prístupu ako celku. Ak napríklad nebude správne pracovať predikcia metriky, tak je možné, že všetky nájdené cesty nebudú vhodné a nebude záležať na tom, ktorú cestu vyberie druhá časť prístupu. To samozrejme platí aj opačným smerom. Druhou nevýhodou môže byť aj väčšia záťaž na samotný centrálny kontrolér, keďže výber každej jednej cesty pre toky je ešte náročnejší ako pri predchádzajúcich prístupoch.

V algoritme č. 5 je zobrazený pseudokód opisujúci kombinovaný prístup k smerovaniu dátových tokov v počítačovej sieti. Keďže tento prístup využíva aj predikciu metrík aj výber cesty, v tomto prípade je potrebné zjednotiť oba prístupy do jedného celku, pričom najprv sa musí vykonať predikcia metrík, na základe

Algoritmus 5 Základný algoritmus pre výber cesty z množiny ciest z predikovaných váh liniek

```

1: procedure MAKE_ROUTING_DECISION(state, new_flow, n)
2:    $s \leftarrow \text{create\_state}(\text{state})$ 
3:    $\text{metric} \leftarrow \text{empty\_set}$ 
4:   for link in  $s$  do
5:      $\text{link\_weight} \leftarrow \text{get\_metric}(\text{link})$ 
6:      $\text{metric.append}(\text{link\_weight})$ 
7:   end for
8:    $\text{paths} \leftarrow \text{select\_n\_paths}(\text{state}, \text{new\_flow}, n)$ 
9:    $s \leftarrow \text{create\_state}(\text{state}, \text{paths})$ 
10:   $\text{route} \leftarrow \text{select\_route}(s)$ 
11:  return  $\text{route}$ 
12: end procedure

```

ktorej sa vytvára množina n ciest. Najprv v riadku 2 sa vytvorí stav reprezentujúci aktuálnu situáciu v počítačovej sieti a v nasledujúcom riadku 3 sa vytvorí prázdna sada metrík. V riadkoch 4 – 7 sa predikujú váhy liniek jednotlivých liniek v počítačovej sieti. Táto časť prebieha rovnako ako pri samostatnom využití predikcie metrík. V riadku 8 sa ale miesto výberu jednej cesty na smerovanie dát nájde n najkratších ciest v počítačovej sieti. Z týchto n ciest sa vytvorí ich reprezentácia stavu v riadku 9. Tento stav sa potom použije na výber najlepšej cesty podľa agenta pre výber cesty, ako je možné vidieť v riadku 10. Táto druhá časť prebieha podobne ako prebieha samostatný výber cesty.

5.4 Metódy strojového učenia

V tejto sekcii budú opísané niektoré metódy strojového učenia, ktoré bude možné použiť pri implementácii navrhovaných prístupov zo sekcie č. 5.3.

Keďže ani v jednej z navrhovaných metód nevieme vopred povedať, ktoré možnosti (akcie) boli správne, a ktoré nesprávne, najvhodnejšou metódou strojového učenia je učenie posilňovaním [41]. Cieľom učenia posilňovaním je nájsť najlepšiu politiku pre dosiahnutie nejakého cieľa. Pri tejto metóde sa agent učí na základe svojich vlastných skúseností v prostredí tak, že sám skúša vykonať rôzne akcie v rôznych stavoch prostredia. Za tieto akcie dostáva agent odmeny, na základe ktorých sa učí, ktoré akcie v ktorom stave sú najlepšie.

Existuje veľké množstvo algoritmov pre tréning agenta s využitím myšlien-

nok učenia posilňovaním. Časť z nich ich ale nie je možné efektívne využiť s navrhovanými prístupmi, keďže buď si vytvárajú pre každý možný stav jeho ohodnotenie, čo je samozrejme nevhodné pre kontinuálny stavový priestor, v ktorom existuje nekonečne veľa možných stavov, alebo sú vhodné pre iné definície akcií.

Pri predikcii metriky bude použitý najmä algoritmus Twin Delayed Deep Deterministic Policy Gradient (TD3) [38] a pre výber cesty bude použitý algoritmus Deep Q Network (DQN) [39], konkrétne jeho vylepšenie Rainbow, a Proximal Policy Optimization (PPO) [42]. Vysvetlenie princípu algoritmov je možné nájsť v prácach, ktoré definujú tieto algoritmy, tieto práce sú spomenuté na začiatku sekcií venovaných jednotlivým algoritmom, ako aj na niektorých internetových stránkach, príklady: ⁵ ⁶.

5.4.1 Algoritmus TD3

Algoritmus TD3 vznikol rozšírením algoritmu DDPG, ktorého cieľom bolo zlepšiť stabilitu pôvodného algoritmu a tým znížiť jeho citlivosť na parametre tréningu. Preto je pred oboznámením sa s algoritmom TD3 vhodné najprv porozumieť algoritmu DDPG.

Algoritmus DDPG, opísaný v práci [43], funguje na princípe toho, že sa učí súčasne politiku, aké akcie sa majú vykonať v ktorom stave, a zároveň odhaduje Q funkciu, funkciu priradzujúcu ohodnotenie jednotlivým stavom. Tento algoritmus sa učí mimo politiky, to znamená, že na učenie môže využívať aj skúsenosti, ktoré nepochádzajú z aktuálnych rozhodnutí politiky samotného agenta. Taktiež je tento algoritmus prispôbený na práce so spojitými akciami, to sú akcie, ktoré nie sú charakterizované len ich vykonaním alebo nevykonaním.

Na učenie Q funkcie vychádza algoritmus z Bellmanovej rovnice pre výpočet ohodnotenia páru stav-akcia. Pre pochopenie práce algoritmu, je potrebné definovať túto rovnicu:

$$Q(s, a) = r(s, a) + \gamma * \max_{a'} Q(s', a'),$$

kde s je aktuálny stav prostredia, s' je nasledujúci stav prostredia, a je akcia, ktorá bola vykonaná agentom v stave s , a' je akcia vykonaná v nasledujúcom stave a γ je parameter, ktorý určuje do akej miery bude vplývať ohodnotenie nasledujúcich stavov na ohodnotenie aktuálneho stavu. Pre opis optimálneho páru stav-akcia, by miesto nejakej ľubovoľnej funkcie $Q(s, a)$ bola použitá optimálna funkcia $Q^*(s, a)$.

⁵<https://spinningup.openai.com/en/latest/index.html>

⁶<https://huggingface.co/>

Keďže ale nie je možné vedieť presnú hodnotu optimálneho ohodnotenia nasledujúcich stavov, je potrebné optimálne hodnoty odhadovať. Preto miesto optimálnej funkcie sa používa jej odhad $Q^\phi(s, a)$. Potom pre zlepšovanie odhadu je možné použiť rovnicu:

$$L(\phi, D) = \begin{cases} (Q_\phi(s, a) - (r + \gamma \max_{a'} Q_\phi(s', a')))^2 & \text{ak nie je epizóda} \\ & \text{ukončená} \\ (Q_\phi(s, a) - r)^2 & \text{ak je epizóda ukončená} \end{cases},$$

kde D je súbor skúseností agenta, ktoré sú v tvare (s, a, r, s', d) , kde d hovorí o ukončení epizódy v danom stave.

Algoritmus tento princíp rozširuje o cieľové estimátory, ktorých parametre sa menia priebežne na princípe:

$$\phi_{\text{targ}} = \rho \phi_{\text{targ}} + (1 - \rho) \phi,$$

kde ϕ sú aktuálne parametre estimátora, používaného na odhad hodnôt páru stav-akcia, ϕ_{targ} sú parametre cieľového estimátora a ρ je parameter určujúci do akej miery majú byť ovplyvnené parametre estimátora ϕ_{targ} aktuálnymi parametrami estimátora ϕ .

Potom je možné vyjadriť chybu funkcie Q_ϕ odhadu páru stav-akcia ako:

$$L(\phi, D) = \begin{cases} (Q_\phi(s, a) - (r + \gamma \max_{a'} Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))))^2 & \text{ak nie je epizóda} \\ & \text{ukončená} \\ (Q_\phi(s, a) - r)^2 & \text{ak je epizóda} \\ & \text{ukončená} \end{cases},$$

kde $\mu_{\theta_{\text{targ}}}$ predstavuje výber akcie cieľovým estimátorom politiky.

Funkcia politiky je estimovaná pomocou estimátora μ_θ . Tak ako pri estimátore funkcie páru stav-akcia, tak aj tento estimátor má svoj cieľový estimátor $\mu_{\theta_{\text{targ}}}$, ktorý vystupuje pri výpočte chyby pre funkciu odhadu páru stav-akcia. V tomto prípade sa taktiež aj podobne upravujú hodnoty parametrov cieľového estimátora:

$$\theta_{\text{targ}} = \rho \theta_{\text{targ}} + (1 - \rho) \theta,$$

kde ρ predstavuje veľkosť vplyvu aktuálnych parametrov estimátora μ_θ na hodnoty parametrov cieľového estimátora $\mu_{\theta_{\text{targ}}}$.

Učenie samotného estimátora funkcie politiky μ_θ je vykonané výpočtom stúpania gradientu pri hľadaní:

$$\max_{a'} Q_\phi(s, \mu_\theta).$$

Algoritmus TD3, opísaný v práci [38], je založený na algoritme DDPG. Hlavné zmeny oproti pôvodnému algoritmu sú:

- zdvojenie estimátora funkcie páru stav-akcia,
- menej častá úprava cieľových estimátorov,
- a pridávanie šumu pri výbere akcií.

Ako vyplýva z prvého bodu, miesto jedného estimátora funkcie $Q_\phi(s, a)$, existujú dva estimátory $Q_{\phi_1}(s, a)$ a $Q_{\phi_2}(s, a)$, a k nim aj ich cieľové estimátory $Q_{\phi_{1\text{targ}}}(s, a)$ a $Q_{\phi_{2\text{targ}}}(s, a)$. Z toho je možné vypočítať chyby pre oba estimátory:

$$L_1(\phi, D) = \begin{cases} (Q_{\phi_1}(s, a) - (r + \gamma \max_{a'} Q_{\phi_{1\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))))^2 & \text{ak nie je epizóda} \\ & \text{ukončená} \\ (Q_{\phi_1}(s, a) - r)^2 & \text{ak je epizóda} \end{cases},$$

$$L_2(\phi, D) = \begin{cases} (Q_{\phi_2}(s, a) - (r + \gamma \max_{a'} Q_{\phi_{2\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))))^2 & \text{ak nie je epizóda} \\ & \text{ukončená} \\ (Q_{\phi_2}(s, a) - r)^2 & \text{ak je epizóda} \end{cases},$$

z ktorých potom je možné vybrať menšiu z oboch hodnôt, ktorú následne je možné použiť ako cieľovú hodnotu.

Druhý bod hovorí len o tom, že nie je potrebné aktualizovať cieľové estimátory funkcie politiky a funkcie hodnoty páru stav-akcia pri každom jednom kroku tréningu, ale je možné znížiť ich frekvenciu, pričom sa dá dosiahnuť takýmto spôsobom lepšiu stabilitu tréningu.

Posledný bod hovorí, že pridaním šumu k hodnote akcií pri výpočte chyby pre estimátor $Q_\phi(s, a)$, sa dá doceliť lepšia stabilita tréningu agenta. Šum sa pridáva k akciám takýmto spôsobom:

$$a(s') = \text{clip}(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), \alpha_{\text{low}}, \alpha_{\text{high}}),$$

kde ϵ predstavuje hodnotu šumu, ktorá je obmedzená pomocou funkcie clip na hodnoty v rozsahu $-c$ až c , α_{low} vyhradzuje najmenšiu hodnotu akcie a α_{high} jej najväčšiu možnú hodnotu.

Podľa poslednej rovnice sa potom upraví výpočet chýb estimátorov funkcie

$Q_{\phi 1}(s, a)$ a $Q_{\phi 2}(s, a)$ na:

$$L_1(\phi, D) = \begin{cases} (Q_{\phi 1}(s, a) - (r + \gamma \max_{a'} Q_{\phi 1_{\text{targ}}}(s', a(s'))))^2 & \text{ak nie je epizóda ukončená} \\ (Q_{\phi 1}(s, a) - r)^2 & \text{ak je epizóda ukončená} \end{cases},$$

$$L_2(\phi, D) = \begin{cases} (Q_{\phi 2}(s, a) - (r + \gamma \max_{a'} Q_{\phi 2_{\text{targ}}}(s', a(s'))))^2 & \text{ak nie je epizóda ukončená} \\ (Q_{\phi 2}(s, a) - r)^2 & \text{ak je epizóda ukončená} \end{cases}.$$

5.4.2 Algoritmus PPO

Ďalším algoritmom, ktorý bude využitý v tejto práci, je algoritmus PPO. Tento algoritmus je opísaný v práci [42].

Cieľom toho algoritmu je zamedziť príliš veľkým zmenám politík, ktoré by mohli spôsobiť náhle zmeny vo výkone agenta. Samotný algoritmus je založený na myšlienke algoritmu Trust Region Policy Optimization (TRPO), pričom ale miesto pomerne náročných výpočtov sa PPO snaží skôr využívať odhady, respektíve menej náročné výpočty.

Na začiatok je dobré povedať, čo je to výhoda. Výhoda je hodnota, ktorá hovorí o koľko je lepšia alebo horšia akcia a v stave s oproti priemernej hodnote všetkých akcií na danom stave. Ako aj vyplýva z predchádzajúcej vety, táto hodnota je záporná, ak akcia a je horšia ako priemerná akcia na danom stave s a je kladná, ak daná akcia je lepšia ako priemerná akcia na tomto stave.

Ako bolo už napísané, pri tomto algoritme je snaha obmedziť veľkosť zmien politík pri jednom kroku učenia. Na docielenie tohto, sa pri tomto algoritme používa funkcia *clip*, ktorá obmedzuje zmeny politík do určitej oblasti hodnôt. Chyba sa pri tomto algoritme vypočíta takto:

$$L_{\text{clip}}(s, a, \theta, \theta_{\text{old}}) = \min(\text{ratio}(s, a)A(s, a), \text{clip}(\text{ratio}(s, a), 1 + \epsilon, 1 - \epsilon)A(s, a)),$$

kde $A(s, a)$ je funkcia, ktorá vypočíta výhodu akcie a v stave s , $\text{ratio}(s, a)$ je hodnota, ktorá udáva pomer nových politík k starým politikám a ϵ je hodnota, ktorá udáva v akom rozmedzí sú povolené zmeny politík v jednom kroku tréningu.

Pomery ratio sa vypočítajú ako:

$$\text{ratio}(s, a) = \frac{\pi_{\theta}(s, a)}{\pi_{\theta_{\text{old}}}(s, a)},$$

kde $\pi_{\theta}(s, a)$ udáva pravdepodobnosť výberu akcie a v stave s pri nových politikách a $\pi_{\theta_{\text{old}}}(s, a)$ udáva pravdepodobnosť výberu akcie a v stave s pri pôvodných

politikách. Ak je hodnota kladná, tak pri novej politike je väčšia pravdepodobnosť, že bude daná akcia vybraná, ak je záporná, tak pri novej politike je menšia pravdepodobnosť výberu danej akcie.

Z výpočtu $L_{\text{clip}}(s, a, \theta, \theta_{\text{old}})$ vieme usúdiť, že táto hodnota vyberá minimálnu hodnotu z dvoch čiastkových výpočtov. Keďže ratio je vždy kladné, tak znamienko závisí len na vypočítanej výhode $A(s, a)$. Ak je výhoda záporná, tak je vždy zvolená najväčšia hodnota z $\text{ratio}(s, a)$ a $1 - \epsilon$, najväčšia záporná hodnota celkovo. Ak je výhoda kladná, tak sa berie najnižšia hodnota medzi $\text{ratio}(s, a)$ a $1 + \epsilon$, najnižšia kladná hodnota celkovo.

Chybu, pre estimátor hodnoty stavu s $V(s)$, sa dá vypočítať ako:

$$L_{\text{value}}(s, r, s', \phi) = (V_{\phi}(s) - (r + \gamma V_{\phi}(s')))^2,$$

kde ϕ sú parametre esimátora pre hodnotovú funkciu stave $V(s)$, γ je parameter, ktorý určuje vplyv nasledujúcich hodnotení stavov na hodnotení tohto stavu, r je odmena agenta v aktuálnom stave a s' je nasledujúci stav prostredia.

Celková funkcia chyby sa dá vyjadriť ako:

$$L(s, a, r, s', \theta, \theta_{\text{old}}, \phi) = L_{\text{clip}}(s, a, \theta, \theta_{\text{old}}) - c_{\text{value}} L_{\text{value}}(s, r, s', \phi) + c_{\text{entropy}} S_{\pi_{\theta}}(s),$$

kde $S_{\pi_{\theta}}(s)$ je vypočítaná entropia v stave s , c_{value} je hodnota určujúca veľkosť ovplyvnenia chyby hodnotovej funkcie na celkovej chybe a c_{entropy} je bonus, ktorý podporuje exploráciu, výber rôznych akcií, agenta v prostredí.

5.4.3 Algoritmus DQN

Algoritmus DQN je jedným z prvých populárnych algoritmov hlbokého učenia posilňovaním. V tejto práci bude využitá jedna z jeho mnohých odvodených variánt, konkrétne sa bude jednať o algoritmus Rainbow, podľa práce [39]. Táto varianta spája do jedného celku niekoľko rôznych modifikácií základného algoritmu DQN.

Pôvodný algoritmus DQN pracuje na princípe odhadu hodnôt všetkých párov stavu a možných akciách na ňom. Tým pádom, algoritmus je založený na estimátorovi funkcie hodnôt párov stav-akcia. Chybu tohto estimátora je možné vypočítať ako:

$$L(s, a, r, s', d) = (y(s', r, d) - Q_{\theta}(s, a))^2,$$

kde s a s' sú aktuálny stav a nasledujúci stav, a je akcia vykonaná agentom v danom stave s , r je dosiahnutá odmena v stave s a d hovorí, či v danom kroku je

ukončená epizóda. Hodnotu $y(s', r, d)$ sa dá získať ako:

$$y(s', r, d) = \begin{cases} r + \gamma \max_{a'} Q_{\theta}(s', a') & \text{ak nie je koniec epizódy} \\ r & \text{ak je koniec epizódy} \end{cases}.$$

Pre zmenšenie vplyvu nadhodnocovania hodnôt páru stav-akcia, sa často využívajú oddelené estimátory pre výber akcie a pre jeho evaluáciu. Tým pádom viem získať nový estimátor funkcie $Q_{\theta_{\text{targ}}}$. Potom hodnotu $y(s', r, d)$ pre chybu estimátora sa dá vypočítať ako:

$$y(s', r, d) = \begin{cases} r + \gamma Q_{\theta_{\text{targ}}}(s', \max_{a'} Q_{\theta}(s', a')) & \text{ak nie je koniec epizódy} \\ r & \text{ak je koniec epizódy} \end{cases}.$$

Tieto estimátory je možné pre ich vzájomné tréningovanie striedať v roli výbercu akcie a evaluátora. Druhou možnosťou je parametre jedného z estimátorov priebežne čiastočne alebo úplne upravovať podľa hodnôt parametrov prvého estimátora.

Ďalšou úpravou pôvodného algoritmu je možné doceliť pomocou úpravy architektúry estimátora, využívaného pre predikciu hodnôt. Kým pôvodne sa priamo predikovala hodnota páru stav-akcia, teraz sa upravuje architektúru estimátora tak, aby samostatne predikoval hodnotu stavu, nezávisle na akcii, taktiež predikoval výhodu každej akcie na danom stave oproti ostatným akciám. Tieto hodnoty potom vieme spojiť do jednej, spoločnej hodnoty, ktorá bude hovoriť o hodnote páru stav-akcia. Tu sa ešte spomína, že je dôležité, aby priemer vypočítaných výhod bol rovný nule, inak takýto prístup nie je korektný.

Ďalším vylepšením fungovania algoritmu je miesto predikcie hodnoty páru stav-akcia, predikovať distribúciu potenciálnych odmien pri výbere akcií v danom stave. Ak je definovaná minimálna možná hodnota ohodnotenia v stave ako $V_{\min}$ a maximálna hodnota ako $V_{\max}$, potom sa dajú rozdeliť hodnoty v rozsahu $V_{\min}$ a $V_{\max}$ do N rôznych kategórií, často v tomto kontexte pomenovaných atómy. Potom vieme pomocou estimátora θ predikovať pravdepodobnosti $p_i(s, a)$ pre akcie a_i v stave s toho, s akou pravdepodobnosťou budú nasledujúce ohodnotenia stavov pri týchto akciách v rozmedzí jednotlivých atómov ako:

$$p_i(s, a) = \exp \theta(s, a).$$

Potom je možné vychádzať z Bellmanovej rovnice modifikovanej pre distribúcie návratov:

$$Z(s, a) =^D r + \gamma Z(s', a'),$$

kde $=^D$ predstavuje operátor pre s' je nasledujúci stav a a' je akcia v nasledujúcom stave. Stratu v tomto prípade počítame pre jeden atóm z_i ako:

$$L(s, a, r, s', d)_i = \sum_{j=1}^N p_i \left(1 - \frac{|z_i - (r + z_j)|}{\Delta_z}\right),$$

kde j je index hodnôt od 1 po počet atómov N , Δ_z je veľkosť jedného atómu, inak povedané veľkosť intervalu medzi z_i a z_{i+1} a p_i je pravdepodobnosť nasledujúcich odmien v daných intervaloch.

Ďalším vylepšením je využitie miesto bežných lineárnych sietí, takzvaných šumivých neurónových sietí, ktoré boli opísané v práci [44]. Cieľom využitia takýchto sietí je zlepšiť preskúvanie prostredia agentom. Hlavným rozdielom oproti bežným lineárnym sieťam je, že kým neuróny v bežných lineárnych sieťach fungujú ako:

$$y = wx + b,$$

kde w je váha priradená vstupom neurónu, x je vstupný vektor a b je dodatočný predsudok, bias. Oproti tomu, pri šumivých lineárnych sieťach, neuróny pracujú ako:

$$y = (\mu_w + \delta_w \odot \epsilon_w)x + \mu_b + \delta_b \odot \epsilon_b,$$

kde $\mu_w, \delta_w, \mu_b, \delta_b$ sú naučiteľné parametre neurónovej siete a ϵ_w, ϵ_b sú náhodné premenné neurónovej siete.

Predposledným vylepšením je používať miesto náhodného výberu skúsenosti pre tréning tie skúsenosti, ktoré sú z nejakého pohľadu zaujímavejšie ako ostatné nazbierané skúsenosti, napríklad na základe rozdielu medzi očakávanou odmenou a získanou odmenou. Na to slúži prioritizovaný buffer skúseností agenta. Jedným z bežných kritérií, podľa ktorých je možné preferovať niektoré skúsenosti pred inými, je ich potenciál z pohľadu ukázania niečoho nového alebo nečakaného, respektíve tie skúsenosti, pri ktorých sa do väčšej miery líši výsledok od predpokladu agenta. Zaujímavosť skúsenosti sa dá napríklad hodnotiť podľa rozdielu medzi predikovanou hodnotou páru stav-akcia, a reálnou hodnotou odmeny, ktorú získal agent od prostredia:

$$td = (r + \gamma \max_{a'} Q(s', a') - Q(s, a))^2.$$

Potom pravdepodobnosť výberu niektorej skúsenosti i sa dá vypočítať ako:

$$P(i) = \frac{p_i^\alpha}{\sum_{k=0}^K p_k^\alpha},$$

kde M je celkový počet skúseností v buffer-i, p_i je priorita vypočítaná pre i -tu skúsenosť, α je parameter, ktorý udáva do akej miery má vplývať priorita na pravdepodobnosť výberu skúsenosti. Prioritu skúsenosti je možné vypočítať ako:

$$p_i = |\text{td}_i| + \epsilon,$$

kde ϵ je malá hodnota zabezpečujúca nenulovosť priority p_i .

Posledným vylepšením pôvodného algoritmu je viac krokové učenie. Rozdielom oproti pôvodnému algoritmu je to, že miesto vypočítavania chyby predikcie hodnoty funkcie stav-akcia s prihliadaním len na odmenu a predikciu nasledujúceho stavu, sa bude prihliadať na viaceré kroky takto:

$$\text{dt}_t = R_t^n + \gamma^n Q(s_{t+n}, a_{t+n}),$$

kde t je aktuálny krok, s_{t+n} a a_{t+n} je stav a akcia o n krokov po aktuálnom kroku, n je počet krokov, na ktoré pri výpočte prihliadame, a R_t^n sa vypočíta ako:

$$R_t^n = \sum_{i=0}^{n-1} \gamma^i r_{t+i},$$

kde r_{t+i} je odmena o i krokov po aktuálnom stave.

Všetky tieto spomínané vylepšenia je možné implementovať do jedného algoritmu, pričom ako celok sa potom takýto algoritmus označuje ako Rainbow.

5.5 Prostredie

Keďže je nepraktické vykonávať tréning a testovanie navrhovaných prístupov pre smerovanie sieťovej komunikácie medzi koncovými zariadeniami na reálnej počítačovej sieti, je potrebné vhodne modelovať prácu takejto počítačovej siete. Na základe takéhoto modelu budú vykonané aj experimenty pre porovnanie dosiahnutých výsledkov jednotlivých smerovacích prístupov.

Pre tieto účely je možné zvoliť už existujúce simulačné prostredia, ako je napríklad NS-3⁷. Výhodu využitia už existujúceho simulačného prostredia je hlavne to, že nie je nutná samostatná implementácia takéhoto prostredia, ako aj väčšia istota správnosti simulácie v takomto prostredí. Ale veľkou nevýhodou sú obmedzenia, ktoré takéto prostredia prinášajú, keďže je nutné stále pracovať len v medziach toho, čo dané prostredie dovoľuje. Taktiež nie je možné väčšinou robiť úpravy prostredia, ako aj zefektívniť samotnú rýchlosť simulácie, napríklad zanedbaním procesov, ktoré síce sú dôležité pre správnu prácu siete, ale nie sú

⁷<https://www.nsnam.org/>

podstatné z pohľadu čistého smerovania a experimentálneho porovnávania rozličných prístupov k smerovaniu. Asi najväčšou nevýhodou takýchto prostredí je ale nutnosť komplexnejších implementácií pre interakciu navrhovaného systému so sieťou.

Práve z týchto dôvodov je pre účely tejto práce naimplementované vlastné prostredie pre modelovanie chodu siete. V tejto sekcii bude daný návrh simulovaného prostredia pre tréning a evaluáciu návrhov opísaný.

5.5.1 Modelované prostredie

Pre účely práce bolo vytvorené modelované prostredie, ktoré umožňuje modelovať zjednodušenú počítačovú sieť a komunikačné toky v nej. Reprezentácia počítačovej siete v simulovanom prostredí je vytváraná na základe jej reprezentácie pomocou dátovej štruktúry, ktorá je vytváraná v navrhovanom systéme.

Pre účely prostredia, celú štruktúru reprezentujúcu topológiu siete, vieme rozdeliť na tri hlavné časti:

- Uzly, reprezentujúce sieťové zariadenia, cez ktoré prechádza komunikácia medzi koncovými bodmi.
- Hrany, reprezentujúce fyzické prepojenia medzi sieťovými zariadeniami.
- Toky, reprezentujúce aktuálne toky dát v sieti.

Na základe týchto častí je možné vytvoriť reprezentáciu siete, ktorá je vhodnejšia na nasledovnú simuláciu práce siete ako nespracované pôvodné dáta.

Za účelom samotnej simulácie práce siete, sú náhodne generované toky dát medzi koncovými zariadeniami siete. Pre zjednodušenie práce siete, samotné koncové zariadenia nie sú modelované, ale sú len abstrahované tak, akoby toky začínali až na zariadení, ktoré dané toky smeruje. K tokom je aj generované množstvo informácií ohľadom ich nárokov na požadované vlastnosti siete, tu môžu patriť požiadavky na oneskorenie, na dostupné pásmo, komunikačnú latenciu a podobne. Okrem týchto hodnôt sa generuje aj interná, náhodná hodnota reprezentujúca koľko časových krokov má daný tok existovať, v tejto práci často spomínaná pod skratkou *TTL*.

Pri každom časovom kroku sa generujú nové toky dát, ako aj sa upravujú stavy uzlov a hrán podľa existujúcich tokov, ktoré cez ne prechádzajú. Podľa počtu tokov dát a ich požiadaviek na zdroje, sú upravované hodnoty dostupnej šírky pásma, počtu prichádzajúcich a odchádzajúcich paketov, počtu chybných paketov a podobne. Tiež je aj dekrementovaná hodnota reprezentujúca počet časo-

vých krokov, koľko má tok dáť existovať v simulovanom prostredí. Ak je dané číslo znížené na nulu, daný tok je odstránený zo simulovaného prostredia.

Prostredie zodpovedá za poskytnutie aktuálneho stavu siete, ktorý slúži ako vstup pre agenta, ako aj za udelenie odmeny agentovi za akciu v prostredí.

5.5.2 Model siete

V tejto sekcii je opísaný spôsob, akým je modelovaná komunikácia v počítačovej sieti. Metóda modelujúca počítačovú sieť je z časti inšpirovaná prácou autorov Kim et. al. [13], kde opisujú nimi navrhnutý model počítačovej siete.

Daný model siete je založený na $M/M/1/K^8$ modelu frontov, ktorý je sám založený na modeli $M/M/1^9$ frontov. Model $M/M/1$ modeluje fronty takým spôsobom, že v ľubovoľnom momente môže prísť požiadavka na server, v našom prípade paket na rozhranie sieťového zariadenia, pričom ak nemôže byť v tom momente spracovaný, tak je zaradený do fronty. V tejto fronte čaká, až kým nemôže byť spracovaný. Fronta v tomto modeli je jednoduchá FIFO (First In First Out) fronta, ktorá ma nekonečnú kapacitu. Model $M/M/1/K$ rozširuje pôvodný model o konečnú veľkosť fronty. To v prípade tejto práci znamená, že ak veľkosť fronty je prekročená, tak ďalšie pakety, ktoré prídu na rozhranie, už nie je možné do fronty zaradiť a sú zahodené.

Počítačovú sieť je možné opísať ako množinu vrcholov (zariadenia) a množinu hrán (linky) medzi nimi. V počítačovej sieti s N zariadeniami a M hranami, sa dá opísať množinu vrcholov ako $V = v_1, v_2 \dots v_N$, a množinu hrán ako $E = e_1, e_2 \dots e_M$. Tým pádom si je možné predstaviť počítačovú sieť ako graf G , pre ktorý platí $G = (V, E)$ a taktiež platí, že pre každé dva uzly grafu G existuje aspoň jedna cesta medzi nimi. Z toho vyplýva, že ak existuje viac ciest medzi dvoma uzlami, tieto cesty môžu mať iné vlastnosti, ako je dĺžka, vyťaženosť a oneskorenie. Preto medzi nimi musí existovať cesta, ktorá bude z pohľadu niektorej zo spomínaných vlastností lepšia ako iné cesty medzi dvoma uzlami v grafe G . Pre každú linku platí, že vie spracovať μ_i paketov, tu sa vychádza z predpokladu, že obe strany linky majú rovnakú kapacitu.

Komunikácia medzi dvoma zariadeniami v sieti sa modeluje pomocou množiny tokov dát f^i , kde i predstavuje číslo, index, toku dát. Pre opis správania toku dát v čase t , je zavedené označenie pre tok dát f_t^i , ktoré opisuje správanie i -teho

⁸<https://real-statistics.com/probability-functions/queueing-theory/m-m-1-k-queueing-model/>

⁹<https://real-statistics.com/probability-functions/queueing-theory/m-m-1-queueing-model/>

toku dát v čase t . Premenná M_t hovorí o počte všetkých tokov v počítačovej sieťi v čase t . Samotný tok je reprezentovaný zdrojovým a cieľovým uzlom, medzi ktorými má komunikácia prebiehať. K tomu je množstvo komunikácie modelované pomocou Poissonových príchodov λ_t^i , pričom i hovorí, ktorému toku hodnota patrí, a t hovorí, pre ktorý časový krok má platnosť. Vlastnosti komunikácie sú následne modelované pomocou očakávaných oneskorení a stratovosti paketov na uzloch, cez ktoré komunikácie dvoch uzlov prechádzajú. V závislosti od použitého navrhovaného prístupu pre smerovania s využitím strojového učenia, použijeme prístupmi vybranú cestu pre komunikačný tok.

Oneskorenie a stratovosť paketov sú modelované pomocou spomínaného modelu frontov M/M/1/K pre jednotlivé linky medzi uzlami. Tu sa dodatočne vychádza z predpokladu, že obe strany linky (hrany) majú rovnakú kapacitu pre spracovanie správ prechádzajúcich na linke. Očakávané oneskorenie na linke e_m v čase t je možné vypočítať ako:

$$D_m = \frac{N_m(t)}{\lambda_m(t)(1 - P_b^m(t))}, \quad (5.2)$$

kde $N_m(t)$ predstavuje množstvo paketov čakajúcich na spracovanie vo fronte pre linku e_m v čase t , $\lambda_m(t)$ je agregované množstvo prichádzajúcich paketov pre všetky toky na hrane a $P_b^m(t)$ je pravdepodobnosť zahodenia paketu v čase t . Hodnotu $P_b^m(t)$ sa dá vypočítať pomocou:

$$P_b^m(t) = \begin{cases} \frac{(1-\rho_m(t))(\rho_m(t))^{K_m}}{1-(\rho_m(t))^{K_m+1}} & \text{ak } \rho_m(t) \neq 1 \\ \frac{1}{K_m+1} & \text{ak } \rho_m(t) = 1 \end{cases} \quad (5.3)$$

kde pre $\rho_m(t)$ platí $\rho_m(t) = \lambda_n(t)/\mu_m(t)$. Pre K_m platí, že to je celkové množstvo komunikácie prechádzajúce cez linku m .

Pre výpočet vyťaženia frontu, počtu paketov vo fronte, sa používa nasledujúci vzorec:

$$N_m(t) = \begin{cases} \frac{\rho_m(t)}{1-\rho_m(t)} - \frac{(K_m+1)(\rho_m(t))^{K_m+1}}{1-(\rho_m(t))^{K_m+1}} & \text{ak } \rho_m(t) < 1 \\ \frac{K_m}{2} & \text{ak } \rho_m(t) = 1 \end{cases} \quad (5.4)$$

Keďže je predpoklad, že propagačné oneskorenie je možné zanedbať, oneskorenie sa modeluje ako:

$$D_p^k(t) = \sum_{n \in p_k} D_n, \quad (5.5)$$

kde p_k predstavuje použitú cestu reprezentovanú množinou liniek, cez ktoré komunikácia prechádza pri smerovaní k -teho toku dát f_k , a D_n je očakávané oneskorenie na n -tej linke cesty.

Celkové priemerné oneskorenie všetkých tokov v čase t je vypočítané ako:

$$D_{avg}^k(t) = \frac{1}{M_t} \sum_{k=1}^{M_t} D_p^k(t), \quad (5.6)$$

kde M_t je počet všetkých tokov komunikácie v sieti v čase t .

Okrem oneskorenia je možné podobne vypočítať očakávanú stratovosť paketov v počítačovej sieti. Stratovosť na jednej linke e_m v čase t sa dá odhadovať ako:

$$L_m(t) = \lambda_m(t) P_b^m(t), \quad (5.7)$$

kde $\lambda_m(t)$ je agregované množstvo prichádzajúcich paketov pre všetky toky na hrane a $P_b^m(t)$ je pravdepodobnosť zahodenia paketov na hrane e_m v čase t na základe vyťaženia fronty danej hrany.

Z predchádzajúceho vzťahu sa dá následne vypočítať celkovú odhadovanú stratu siete ako:

$$L_{total}(t) = \sum_{m=1}^M \lambda_m(t) P_b^m(t), \quad (5.8)$$

kde M je celkový počet všetkých hrán počítačovej siete.

Taktiež predpokladáme existenciu aspoň jednej cesty p_{ij} medzi akoukoľvek dvojicou uzlov v_i a v_j , pričom platí $i \neq j$. Každú cestu p_{ij} sa dá jednoznačne opísať množinou hrán, ktorou je tvorená, t.j. $p_{ij} = e_1^{ij} \dots e_n^{ij}$, kde $0 < i \leq n$, pričom n je počet všetkých hrán, ktorými je cesta p_{ij} tvorená, $n = |p_{ij}|$. Množinu všetkých možných ciest medzi uzlami v_i a v_j budeme označovať ako P_{ij} , pričom n_{path} bude označovať počet všetkých rozličných ciest medzi uzlami v_i a v_j , $n_{path} = |P_{ij}|$, a množina P_{ij} je definovaná ako $P_{ij} = p_{ij}^1 \dots p_{ij}^n$, pričom $0 < l \leq n$. Pre každú z ciest p_{ij}^u platí, že musí existovať aspoň jedna hrana e_u , pre ktorú platí, že $e_u \in p_{ij}^u$ a zároveň $e_u \notin p_{ij}^u$. Najkratšia cesta p_{ij}^* medzi uzlami v_i a v_j je taká cesta z množiny ciest P_{ij} , pre ktorú sa dá vypočítať najnižšia suma váh liniek W_{ij}^l , pričom

$$W_{ij}^l = \sum_{k=0}^{|p_{ij}^l|} w_k^{ij_l}, \quad (5.9)$$

kde $|p_{ij}^l|$ je počet hrán medzi uzlami v_i a v_j pri použití cesty p_{ij}^l a $w_k^{ij_l}$ je váha linky priradená k -tej hrane l -tej cesty medzi uzlami v_i a v_j . Následne p_{ij}^* je možné vypočítať ako

$$p_{ij}^* = p_{ij}^q, \quad (5.10)$$

kde q je index najkratšej cesty pri zohľadňovaní váh liniek a index q sa vypočíta ako

$$q = \arg \min_{l=0}^{|P_{ij}|} W_{ij}^l, \quad (5.11)$$

kde funkcia $\arg \min_{l=0}^{|P_{ij}|} W_{ij}^l$ vracia index cesty s minimálnou hodnotou W_{ij}^l .

5.5.3 Akcie agenta

V tejto sekcii budú opísané akcie, ktoré agenti môžu v prostredí vykonávať. Akcie agentov závisia od použitého návrhu pre smerovanie s využitím strojového učenia, podrobnejší opis návrhov pre smerovania s využitím strojového učenia je v sekcii 5.3. Celkovo z návrhov vyplýva, že je potrebné definovať akcie agentov v prípade predikcie metriky a výbere cesty.

Akcie agenta pri predikcii metriky

V prípade predikcie metriky, akciou agenta je predikovaná metrika hrany (linky) medzi uzlami (zariadeniami) v sieťovej topológii. Ako vyplýva z návrhu prístupu, v tomto prípade sú definované dve možnosti:

- výpočet metriky jednej linky v jednom kroku predikcie,
- súčasný výpočet metriky všetkých liniek v jednom kroku predikcie.

V prvom prípade je akcia agenta reprezentovaná jednou hodnotou váhy linky w_i , ktorá prislúcha i -tej linke počítačovej siete. Z toho vyplýva, že pre výpočet metriky všetkých n liniek v topológii, je potrebné opakovať predikciu n -krát, pričom každú predikciu je potrebné vykonať so stavom reprezentujúcim i -tú linku, kde $0 < i \leq n$.

V druhom prípade agent predikuje metriky všetkých liniek v jednom kroku. To znamená, že akciou agenta je vektor W hodnôt, pričom daný vektor obsahuje n hodnôt, kde každá hodnota w_i , i je v rozsahu 1 až n a $W = w_1, w_2 \dots w_n$, reprezentuje práve jednu linku medzi zariadeniami v topológii siete.

Akcie agenta pri výbere cesty

V prípade výberu cesty z množiny najkratších ciest je využitá len jedna reprezentácia akcie agenta. Agent predikuje najlepšiu akciu pomocou vektora ohodnotení stavov alebo distribúcie pravdepodobnosti výberu akcie. Následne sa vyberá akcia s najväčšou predikovanou hodnotou alebo distribúcie pravdepodobnosti výberu akcie. Táto akcia je reprezentovaná pomocou čísla, indexu, akcie, ktorý hovorí, koľká najkratšia cesta má byť použitá pre smerovanie toku dát.

Agent vracia vektor n hodnôt a_i pre každú akciu, kde $0 < i < n$, potenciálnu cestu, z ktorých agent vyberá. Hodnota n predstavuje počet ciest všetkých možných ciest a i je index cesty. Podľa predikovaných ohodnotení, distribúcie pravdepodobností, sa vracia index i , pri ktorom je najvyššia hodnota p_i .

5.5.4 Reprezentácia stavu

Aktuálny stav počítačovej siete môže byť reprezentovaný viacerými spôsobmi podľa architektúry agenta a na základe povahy vykonávaných experimentov. Rôzne návrhy využívajú rôzne reprezentácie stavu siete podľa toho, čo potrebujú k svojej správnej činnosti. Taktiež každý návrh je experimentálne vyskúšaný s rôznou informáciou poskytnutou v opise aktuálneho stavu siete. Reprezentácie stavu sa dá rozdeliť do dvoch hlavných skupín:

- Stav pre výpočet metriky.
- Stav pre výber cesty.

Obe kategórie stavov sa dajú ešte navyše rozdeliť na prípady, keď chceme využiť rovnako dimenzionálnu reprezentáciu stavu pre viaceré topológie, alebo v prípade, keď každá topológia môže mať rôzne dimenzionálnu reprezentáciu.

Reprezentácia stavu pre predikciu metriky

Najjednoduchšou možnosťou reprezentácie stavu siete v tomto prípade je využitie matice susednosti grafu, kde spolu indexy na osiach x a y reprezentujú uzly a nenulové hodnoty na pozícii $[x, y, z]$, existenciu hrany medzi danými uzlami a hodnota v bode z predstavuje jednu z vlastností hrany alebo susedných uzlov. Výhodou takéhoto prístupu je možnosť poskytnúť väčšie množstvo informácií agentovi pre jeho rozhodnutie a možnosť vypočítavať metriky všetkých hrán v tom istom momente. V tomto prípade ale nie je možné jednoducho použiť rovnako rozmernú reprezentáciu v prípade viacerých topológií.

Druhým spôsobom pre reprezentáciu stavu je vytvorenie reprezentácie stavu len za pomoci informácie ku konkrétnej jednej hrane. Takýmto spôsobom vieme získať vektor hodnôt, kde hodnoty reprezentujú nejakú vlastnosť hrany. V tomto prípade sa dá považovať za veľkú výhodu to, že nie sú nutné žiadne úpravy pre využitie rovnakej reprezentácie pre akúkoľvek topológiu. Za nevýhodu sa dá považovať nutnosť separátnej predikcie metriky pre každú hranu a taktiež nemožnosť, respektíve väčšia komplexnosť, pre poskytnutie dodatočnej informácie o stave siete ako celku.

Tretím spôsobom reprezentácie stavu pre predikciu metriky je využitie informácie o jednej hrane, ako je opísané v prípade predchádzajúceho spôsobu, pričom ale miesto toho, aby sa používal len stav jednej linky, budú spájané viaceré reprezentácie liniek do jedného stavu. Tým sa získa dvojrozmerná matica $x \times y$, kde jeden rozmer určuje počet liniek v topológii počítačovej siete, $x = |E|$, a druhý

rozmer y určuje počet vlastností asociovaných s jednou linkou. Takouto reprezentáciou stavu je následne možné predikovať metriky všetkých liniek súčasne.

Z pohľadu hodnôt, ktoré sa používajú na reprezentáciu stavu hrany, je možné použiť dáta ako

- doterajšia stratovosť paketov na linke,
- vyťaženosť linky,
- maximálna kapacita linky,
- aktuálne oneskorenie na danej linke,
- počet paketov prechádzajúcich linkou,
- počet tokov využívajúcich danú linku.

Rôzne kombinácie týchto dát sa dajú používať na reprezentáciu rôznych stavov siete. Tieto dáta môže byť nutné aj upravovať, hlavne škálovať, keďže v svojom pôvodnom tvare sú dané v rôznych jednotkách.

Pri experimentoch je potrebné zistiť najvhodnejšiu reprezentáciu stavu pre prácu agenta, ako aj to, či je nutné dáta upravovať pre lepšiu prácu agenta.

Reprezentácia stavu pre výber cesty

Pre výber cesty je reprezentovaný stav ako dvojrozmerná matica, kde os x reprezentuje rôzne možné cesty a os y reprezentuje vlastnosti daných ciest. Keďže nie je možné vyhodnocovať každú jednu možnú cestu, počet ciest reprezentovaných týmto stavom je obmedzený na maximálny počet n ciest, z ktorých agent vyberá. Takáto reprezentácia stavu vychádza z práce [19].

Vlastnosti ciest sú reprezentované pomocou dát, ako:

- doterajšia stratovosť paketov na linkách,
- vyťaženosť liniek,
- maximálna kapacita liniek,
- aktuálne oneskorenie na linkách,
- počet paketov prechádzajúcich linkami,
- počet tokov využívajúcich linky.

Pričom na reprezentáciu stavu sa využívajú všeobecne priemerné, maximálne alebo minimálne z daných hodnôt medzi linkami, ktoré by boli využité pri výbere konkrétnej cesty medzi dvoma uzlami.

Výhoda takéhoto prístupu k reprezentácii stavu je tá, že pre akúkoľvek topológiu je vždy možné využiť rovnakú reprezentáciu a dimenzionalita tejto reprezentácie závisí len na počte ciest, z ktorých agent vyberá, a na množine parametrov, ktoré sú poskytované agentovi.

Tieto dáta je, rovnako ako aj v prípade reprezentácie stavu pre predikciu metriky, možné škálovať, aby dosahované hodnoty boli na rovnakej stupnici. Taktiež je potrebné experimentálne zistiť, aká množina dát je najvhodnejšia na reprezentáciu stavu pre agenta, ako aj možnosť škálovania dát.

5.5.5 Reprezentácia odmeny

Odmena dosiahnutá agentom môže byť, tak ako reprezentácia stavu, vypočítavaná rôznym spôsobom v tomto prípade hlavne podľa povahy experimentov a agenta.

Všeobecne, podľa rozsahu predikcií je možné odmeny deliť na odmeny, ktoré hodnotia správanie agenta pri voľbe vplývajúcej na hrany lokálne, a na tie, ktoré vplývajú na celkovú prácu celej siete. V prvom prípade sa dá považovať za postačujúce pozerať na vplyv akcie agenta na hranu a úplne zanedbávať celkový stav siete. V druhom prípade je potrebné zhodnotiť stav siete ako celku.

Pri hodnotení stavu, či už lokálnej hrany alebo počítačovej siete ako celku, je možné vyberať z veľkého množstva merateľných ukazovateľov (vlastností). Medzi najčastejšie používané vlastnosti pri vykonaných experimentoch patria dostupná voľná šírka pásma, stratovosť komunikácie na hrane, množstvo tokov prechádzajúcich cez hranu alebo uzol, oneskorenie asociované s tokmi prechádzajúcimi sieťou. Na základe rôznych kombinácií vlastností je možné vytvárať rôzne typy odmien.

Reprezentácia odmeny pri predikcii metriky

V prípade predikcie metriky je nutné rozdeliť odmenu pre prípad predikcie jednej linky a pre prípad predikcie pre všetky hrany topológie v jednom kroku.

V prvom prípade, je odmena reprezentovaná ako pomer odmeny za celkový stav topológie, hodnota r_g , a stav konkrétnej predikovanej linky, hodnota r_l , daný pomer nech je definovaný konštantou α . Potom odmeňovacia funkcia môže byť

definovaná ako:

$$r = \alpha * r_g + (\alpha - 1) * r_l,$$

pričom r je finálna odmena a konštanta α nadobúda hodnoty od 0 do 1. Z toho aj vyplýva, že v extrémnych prípadoch, α rovné 0 alebo 1, je možné vplývať na rozhodnutia agenta výlučne na základe lokálneho stavu hrán alebo výlučne celkovým stavom siete:

$$r = r_l.$$

V druhom prípade je odmena definovaná len ako funkcia celkového stavu siete:

$$r = r_g.$$

Odmeny za celkový stav siete r_g a lokálny stav na linke r_l je možné vypočítať z rôznych sledovaných ukazovateľov siete. V tejto práci bude čiastková odmena definovaná ako:

$$r_p = \beta * (1 - pl) + (\beta - 1) * (1 - d),$$

kde pl je stratovosť paketov, d je oneskorenie paketov a konštanta β vplýva na pomer medzi týmito dvoma zložkami odmeny na celkovú dosiahnutú odmenu. Pričom:

$$r_g = r_{p_g}$$

a

$$r_l = r_{p_l}.$$

Samotný význam pl a d sa mierne mení v prípade lokálnej linky a celkového stavu siete. V prípade lokálnej linky pl predstavuje priemernú stratovosť paketov pre toky prechádzajúce danou linkou a d je priemerné oneskorenie paketov pre toky prechádzajúce danou linkou. Pri celkovom stave siete je pl priemerná stratovosť paketov pre všetky toky existujúce v danom momente v topológii a d je priemerné oneskorenie paketov pre dané toky. Z toho vyplýva, že pri výpočte r_{p_g} sa využívajú hodnoty vypočítané pre hodnotenie celkového stavu topológie a pre r_{p_l} hodnoty vypočítané pre hodnotenie konkrétnej linky.

V druhom prípade je, to jest, v prípade súčasnej predikcie všetkých váh liniek, je odmena taktiež rozdelená na dve časti, pričom,

$$r = \alpha * r_g + (\alpha - 1) * r_f.$$

V tomto prípade sa r_g počíta rovnako ako v prípade predikcie každej váhy linky samostatne. Rozdiel je vo výpočte druhej časti odmeny, v tomto prípade pôjde o odmenu zameranú na vlastnosti liniek r_f , cez ktoré bude smerovaný nový tok

dát. Aj v tomto prípade je parciálna odmena vypočítaná z dvoch častí, pl a d . Tieto časti sa v tomto prípade počítajú z aktuálnych vlastností stratovosti a oneskorenia liniek, cez ktoré najnovší dátový tok prechádza, pričom sa berie do úvahy priemerná hodnota vlastností všetkých liniek, cez ktoré posledný dátový tok prechádza.

Reprezentácia odmeny pri výbere cesty

Odmeňovacia funkcia pri výbere cesty je podobná odmeňovacej funkcii použitej pri predikcii metriky agentom. V tomto prípade rovnako vychádzame z dvoch zložiek odmeny, ohodnotenia stavu celej siete r_g a ohodnotenia stavu vybranej cesty r_l , tým taktiež dostaneme rovnicu:

$$r = \gamma * r_g + (\gamma - 1) * r_l,$$

kde γ je taktiež pomer vplyvu týchto dvoch zložiek na celkovú odmenu agenta, pričom hodnota γ sa pohybuje medzi 0 až 1.

Taktiež vychádzame z predpokladu, že platí

$$r_g = r_{p_g}$$

a

$$r_l = r_{p_l}.$$

Pričom r_{p_g} a r_{p_l} sa dá vypočítať z:

$$r_p = \beta * (1 - pl) + (\beta - 1) * (1 - d),$$

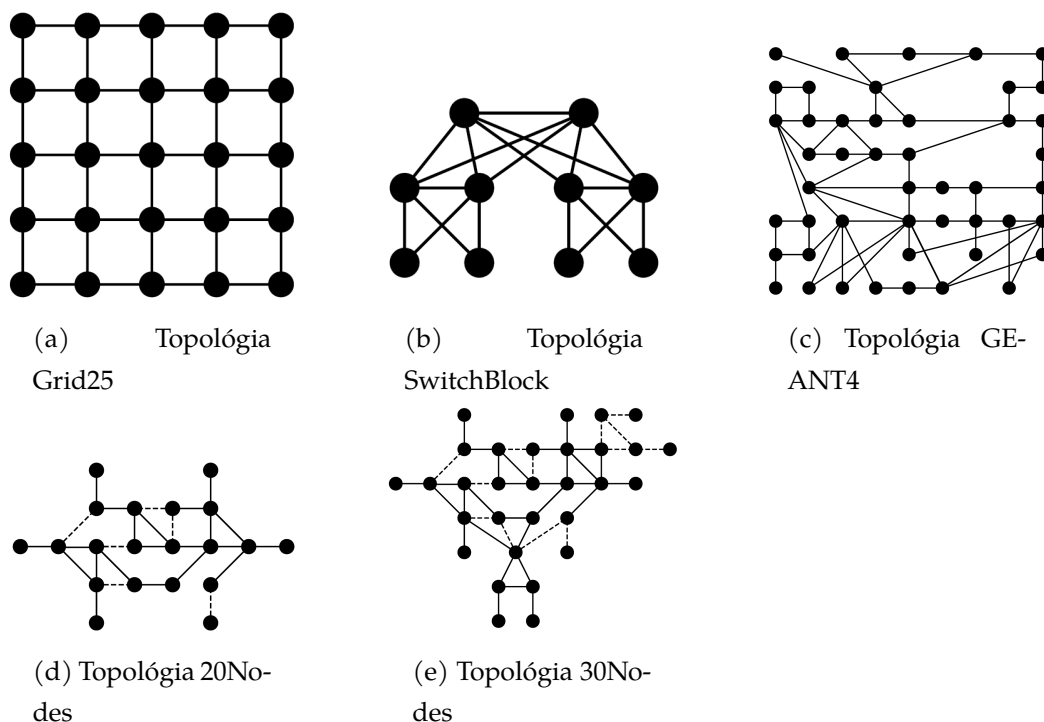
kde pl a d reprezentuje buď stav vybranej cesty, to je prípad r_{p_l} , alebo stav celej siete, v prípade r_{p_g} .

5.5.6 Experimentálne topológie

Pre tréning agentov a modelov je potrebné, okrem simulovaného prostredia, navrhnuť aj topológie počítačových sietí, z ktorých by prostredie vychádzalo. Za týmto účelom bolo navrhnutých niekoľko topológií o rôznych veľkostiach z pohľadu množstva uzlov a hrán. Taktiež boli využité aj topológie opísané v prácach, spomínaných v kapitole č. 3 a relevantných k smerovaniu komunikácie v infraštruktúrach prepájaných fyzickým médiom, ako aj tie, ktoré nie sú v nich opísané, ale sú spomínané v kontexte porovnávania výsledkov navrhovaných prístupov.

Tabuľka 5.1: Sumarizácia navrhovaných topológií

Topológia	Počet uzlov	Počet hrán
Veľmi malá topológia	10	15
Malá topológia	20	27
Stredná topológia	30	43
Switch Block	10	19
Grid25	25	40
GEANT4	47	66



Obrázok 5.2: Topológie využité v experimentoch

Za účelom posudzovania správnosti fungovania navrhovaných prístupov je potrebné experimentálne využívať navrhované metódy pri ich uplatnení pri smerovaní tokov komunikácie v počítačovej sieti. Za týmto účelom je potrebné využiť tieto spomínané topológie pre porovnanie vlastností počítačovej siete, ako aj vlastností komunikačných tokov, pri využití opísaných prístupov a ich porovnaní s inými možnosťami realizácie smerovania v sieti.

Použité návrhy topológií sietí sú sumarizované v tabuľke č. 5.1 a zobrazené na obrázku č. 5.2. Použité topológie sa do značnej miery líšia ich počtom uzlov (smerujúcich sieťových zariadení) a počtom hrán (fyzických prepojení medzi nimi). Podľa množstva uzlov a hrán dostali prvé tri navrhované topológie svoje

názvy. Veľmi malá topológia pozostáva iba z 10 uzlov a 15 prepojení medzi nimi. Táto topológia bude slúžiť primárne len pre skúšobné testovanie funkčnosti testovaných prístupov a nebude využitá pre porovnávanie dosiahnutých výsledkov odskúšaných prístupov. Pomocou nasledujúcich dvoch topológií by už bolo možné porovnávať schopnosť práce jednotlivých prístupov v pomerne malých sieťach. V tomto prípade v sieťach s 20 uzlami a 27 prepojeniami medzi nimi, ktorú nazývame 20Node, a s 30 uzlami a 43 prepojeniami medzi nimi, ktorú nazývame 30Node. Tieto dve topológie sú zobrazené na obrázkoch č. 5.2d a č. 5.2e. Tieto topológie sú taktiež odlišné v tom, že majú definované hrany s menšou a väčšou priepustnosťou. To je zobrazené na obrázkoch na základe odlišnej čiary spájajúcej jednotlivé uzly, pričom celá čiara reprezentuje väčšiu priepustnosť a čiarkovaná menšiu.

Ostatné topológie sú bežne používané v prácach, ktoré boli spomínané v kapitole č. 2. Prvá z nich má názov SwitchBlock, ktorá je zobrazená na obrázku č. 5.2b. Táto topológia je relatívne malá, má len 10 uzlov a 19 hrán, predstavuje ale jednu z tradičných architektúr, podľa ktorých sa tvoria návrhy počítačových sietí. Druhá má názov Grid25 a je zobrazená na obrázku č. 5.2a. Grid25, ako aj vyplýva z názvu, je mriežka 25 uzlov (päť na päť), ktoré sú prepojené so svojimi susedmi, z toho vyplýva, že má 40 hrán. Posledná topológia má názov GEANT4 a je na obrázku č. 5.2c. Táto topológia je najväčšia so 47 uzlami a 66 hranami.

6 Opis implementácie

Táto kapitola obsahuje opis implementácie návrhu riešenia smerovania, ktorý bol opísaný v kapitole č. 5. Keďže táto práca sa venuje smerovaniu dátových tokov v počítačových sieťach s využitím strojového učenia, v tejto práci je navrhovaný systém implementovaný v tej miere, ktorá je potrebná pre realizáciu experimentov s využitím navrhovaných prístupov k smerovaniu dátových tokov s využitím strojového učenia a vybraných tradičných metód smerovania pre vykonanie porovnania výkonu navrhovaných prístupov k smerovaniu s tradičnými metódami.

Členenie kapitoly je nasledovné: v sekcii č. 6.2 je opísaná implementácia sieťového modelu, v sekcii č. 6.1 je opísaná implementácia prostredia, vrátane reprezentácie stavu a odmeny, a v sekcii č. 6.3 sú opísané implementácie navrhovaných prístupov k smerovaniu ako aj realizácia tradičných metód smerovania. Opis priebehu samotných experimentov je opísaný v kapitole č. 7.

6.1 Implementácia prostredia

Prostredie je implementované na základe opísaného návrhu v sekcii č. 5.5. Kým v spomínanej sekcii je opísaná teoretická predstava a návrh fungovania prostredia, v tejto sekcii bude opísaná realizácia daného návrhu.

V jednotlivých podsekcích sú opísané konkrétne implementácie častí prostredia. Niektoré časti sú implementované niekoľkými spôsobmi, buď z dôvodu experimentovania s rôznymi možnými realizáciami tej istej funkčnej časti, napríklad rôzne implementácie možnej reprezentácie stavu, odmeny alebo realizácie tréningovej slučky, alebo preto, lebo rôzne prístupy k smerovaniu s využitím strojového učenia si vyžadujú odlišné implementácie tej istej funkcionality z dôvodu nekompatibility prístupov. Tieto rozdiely budú v relevantných častiach podsekcí bližšie spomenuté.

6.1.1 Prostredie

Pre účely sprostredkovania modelovania správania siete vzhľadom k interakcii navrhovaného systému s počítačovou sieťou, bolo navrhnuté prostredie, ktoré bude realizovať interakciu medzi prostredím a navrhovaným systémom. Interakcia s prostredím funguje nasledovne, pričom vychádza z predpokladu, že v každom kroku je potrebné pridať do počítačovej siete nový dátový tok:

1. Vznikne požiadavka na nový dátový tok.
2. Nájde sa cesta pre nový dátový tok, ktorá bude reprezentovať smerovaciu politiku pre daný dátový tok.
3. Politika sa aplikuje na relevantných manažovaných uzloch.
4. Modeluje sa správanie siete za daný časový krok, pričom, keďže v predchádzajúcom časovom kroku ešte nový dátový tok neexistoval, nebude v tomto časovom kroku braný do úvahy pri modelovaní správania siete.
5. Na základe nového stavu siete sa vypočíta odmena a vytvorí sa nová reprezentácia stavu počítačovej siete.

Tieto kroky bežia v cykle, pričom dĺžka cyklu je definovaná dĺžkou experimentu, alebo, v reálnom živote, pokým beží navrhovaný systém.

Realizáciou týchto krokov je implementácia prostredia ako celku, pričom k prostrediu je vytvorená trieda *Environment*, ktorá definuje základný súbor funkcií potrebných pre realizáciu všetkých vyššie spomínaných krokov. Samotné kroky sú definované a realizované v metódach, a to krok č. 1 je realizovaný pomocou metódy *create_flow*, kým za nájdenie novej smerovacej politiky zodpovedá prostredie, za jej aplikáciu už zodpovedá. To sa realizuje v metóde *step*, pričom v čiastkových funkciách a metódach *update_flow*, *get_state* a *get_reward* sa upravuje aktuálna politika priradená danému dátovému toku. Druhé dve metódy zodpovedajú za sprostredkovanie získania odmeny a reprezentácie stavu počítačovej siete. Posledné dve metódy len volajú priradené implementácie pre reprezentáciu stavu a výpočet odmeny, ktoré boli získané pri inicializácii prostredia na základe poskytnutých parametrov. Tieto implementácie budú bližšie spomenuté v podsekciiach č. 6.1.2 a č. 6.1.3. Za modelovanie správania siete zodpovedá statická funkcia *check_flows* na základe poskytnutej topológie v parametroch funkcie pri jej volaní.

Pre túto prácu bolo implementovaných päť verzií prostredia, *EnvironmentMetric*, *EnvironmentMetricALL*, *EnvironmentPath*, *EnvironmentPathPPO* a *Environment-*

Comb, pre každý z navrhovaných prístupov, keďže každý prístup si vyžaduje mierne odlišnú implementáciu jednotlivých spomínaných metód. Kým v prípade implementácie vhodnej pre agenta predikujúceho metriku, či pre všetky linky súčasne alebo samostatne, je potrebné nájsť novú cestu na základe poskytnutých predikovaných metrík, v prípade ostatných implementácií je potrebné len upraviť smerovaciu politiku priamo na základe poskytnutej finálnej cesty. Ďalším rozdielom je zoznam podporovaných možných reprezentácií stavu počítačovej siete. Kým niektoré reprezentácie sú vhodné pre konkrétneho agenta, všeobecne nemusia byť vhodné pre druhého. Z tohto dôvodu má každá implementácia len obmedzené množstvo podporovaných reprezentácií stavov a odmien. Taktiež je potrebné zohľadniť to, že pri využití kombinovaného prístupu je potrebné zabezpečiť prístup k dvojici reprezentácií stavov a odmien. Tým pádom nie je možné použiť jednu implementáciu prostredia so smerovacím prístupom, pre ktorý nie je implementovaná.

6.1.2 Reprezentácie stavu

Tak, ako sú opísané v sekcii č. 5.5, boli implementované reprezentácie stavu pre navrhovaný systém. Pre zjednodušenie a zjednotenie rozhrania potrebného pre získanie stavu siete, bolo využité jednotné rozhranie pre získanie stavu v prípade agenta pre výber cesty a pre agenta pre predikciu metriky. Takýmto spôsobom vznikli len jednotné použiteľné rozhrania, pričom každá implementácia stavu agenta využíva vždy rovnaké rozhranie.

Každá implementácia reprezentácie stavu bola implementovaná ako samostatná trieda, pričom táto trieda dedí od všeobecnej triedy, v ktorej sú definované metódy, ktoré musia byť implementované pre každú novú triedu. Každá takáto implementácia musí obsahovať vlastnú implementáciu metód *get_observation_space*, *get_action_space* a *get_state*. Metóda *get_observation_space* slúži na získanie dimenzionality vráteného stavu, *get_action_space* pre získanie dimenzionality akčného priestoru a *get_state* slúži na vrátenie samotného stavu. Kým prvé dve metódy majú rovnakú signatúru v prípade oboch agentov, posledná metóda vyžaduje parameter naviac. Spoločným parametrom je inštancia triedy *Topology*, pričom pri využití agenta pre výber cesty je navyše potrebné poskytnúť parameter pre množinu ciest. Každá implementácia môže taktiež navyše obsahovať atribúty.

Rôzne implementácie môžu vytvárať požadovanú reprezentáciu stavu rôzne, pričom sa môžu používať rôzne kombinácie vlastností spomínaných v sekcii č. 5.5, ktoré môžu byť, ale aj nemusia byť, škálované alebo obmedzené do určitého intervalu. V prípade predikcie metriky sa tieto vlastnosti vzťahujú prevažne na

konkrétnu predikovanú linku, zatiaľ čo pri výbere cesty sa tieto parametre skôr vzťahujú na konkrétnu cestu.

6.1.3 Reprezentácie odmeny

Tak, ako boli opísané v sekcii č. 5.5, ako boli opísané a implementované stavy, tak isto boli implementované aj reprezentácie odmeny. Rovnako boli taktiež použité rozhrania pre rôzne implementácie odmeny pre rovnakého agenta. Takže takýmto spôsobom vzniklo taktiež jednotné použiteľné rozhranie, pričom každý agent má len jednu možnú implementáciu rozhrania.

V tomto prípade, na rozdiel od reprezentácie stavu, potrebujeme len jednu metódu, a to je *reward_func*, ktorá musí existovať v každej implementácii odmeny agentov. Metóda *reward_func* môže mať ale rozličnú signatúru v prípade agentov predikujúcich metriku a vyberajúcich cestu, v závislosti od metódy výpočtu odmeny. Kým oba agenti potrebujú parameter *Topology*, len agent pre výber cesty môže potrebovať taktiež aj cestu, ktorá bola zvolená pre smerovanie posledného dátového toku, respektíve cesty, ktoré boli na výber. Taktiež môže obsahovať každá implementácia dodatočné metódy alebo atribúty, ktoré sú potrebné pre konkrétnu implementáciu odmeny.

Tak, ako aj v prípade implementácií stavov, aj implementácie odmeny môžu používať rôzne vlastnosti pri výpočte odmeny, ako bolo spomínané v sekcii č. 5.5. Tieto vlastnosti môžu byť zohľadňované v rôznych pomeroch a môžu byť používané v rôznych kombináciách, pričom pomery vplyvu konkrétnych parametrov na celkovú odmenu môže byť upravované pomocou špecifických atribútov, ktoré sú implementované len pre konkrétne odmeny.

6.1.4 Topológia

Topológie tvoria dôležitú súčasť prostredia, v ktorom sa vykonávajú experimenty. Všeobecne boli použité topológie už spomenuté v podsekcii č. 5.5.6, kde sú opísané z pohľadu kompozície a ich vlastností. V tejto podsekcii sú opísané vlastnosti implementácie topológie, ktorá je vytvorená a použitá pre účely tejto práce.

Najprv je potrebné opísať formát dátovej štruktúry, z ktorej je vytvorená reprezentácia topológie počítačovej siete, a ktorá slúži aj pre opis aktuálneho stavu počítačovej siete. Táto dátová štruktúra je zobrazená v nasledujúcich fragmentoch kódu. Prvá časť dátovej štruktúry je zobrazená v kóde č. 6.1.

Výpis 6.1: Dátová štruktúra topológie

1 {

```

2  "nodes": [],
3  "edges": []
4  }

```

Dátová štruktúra obsahuje dva hlavné listy, a to zoznam uzlov a zoznam hrán. Každý z týchto zoznamov obsahuje dátové štruktúry, pričom uzly sú vyjadrené dátovou štruktúrou vyjadrenou v kóde č. 6.2 a hrany v kóde č. 6.4

Výpis 6.2: Dátová štruktúra uzla

```

1  {
2    "serial_number": "serial_number",
3    "node_id": "node_id",
4    "hostname": "name_of_node",
5    "interfaces": [],
6    "cpu_utilization": {
7        "5_second": "percentage_of_utilization",
8        "1_minute": "percentage_of_utilization",
9        "5_minute": "percentage_of_utilization"
10   },
11   "memory_utilization": {
12       "total": "memory_usage_mb",
13       "used": "memory_usage_mb",
14       "free": "memory_usage_mb"
15   },
16   "static_routes": [],
17   "existing_flows": [],
18   "running_config": "output_of_show_run_command"
19 }

```

Každý uzol je reprezentovaný dátovou štruktúrou, ktorá obsahuje najdôležitejšie informácie o ňom. Ako je zobrazené v kóde č. 6.2, táto dátová štruktúra obsahuje sériové číslo reálneho zariadenia reprezentujúceho daný uzol, jeho identifikátor (*node_id*), ktorý daný uzol jednoznačne identifikuje v počítačovej sieti, zoznam rozhraní, kde každé rozhranie je definované pomocou dátovej štruktúry, ktorá je zobrazená v kóde č. 6.3, informácie o vyťaženosti procesora daného uzla v rôznych časových intervaloch, informácia o vyťaženosti pamäte uzla, a taktiež zoznam všetkých ciest, ktoré aktuálne sú nastavené na uzle, zoznam dátových tokov, ktoré cez daný uzol tečú a bežiacu konfiguráciu uzla. Zoznam statických ciest v tejto dátovej štruktúre reprezentuje aktuálne smerovacie politiky, ktoré sú

aktívne na danom uzle.

Výpis 6.3: Dátová štruktúra rozhrania

```

1 {
2   "name": "name_of_interface",
3   "description": "description",
4   "ip_address": "ip_address_of_interface",
5   "subnet_mask": "subnet_mask",
6   "prefix_length": "prefix_length",
7   "mac_address": "mac_address",
8   "status": "status_of_interface",
9   "line_protocol": "status_of_protocol",
10  "speed": "interface_speed",
11  "bandwidth": "interface_bandwidth",
12  "transport_information": {
13    "bandwidth": "interface_bandwidth",
14    "delay": "delay",
15    "input_packets": "num_of_input_packets",
16    "input_bytes": "num_of_input_bytes",
17    "output_packets": "num_of_output_packets",
18    "output_bytes": "num_of_output_bytes",
19    "input_error": "num_of_input_errors",
20    "output_error": "num_of_output_errors"
21  },
22  "neighbor": {
23    "neighbor_name": "neighbour_device_name",
24    "neighbor_ip_address": "neighbour_ip_address",
25    "neighbor_interface": "name_neighbour_interface",
26    "neighbor_mac_address": "neighbour_mac_address"
27  }
28 }
```

V kóde č. 6.3 je vyobrazená dátová štruktúra reprezentujúca jednotlivé rozhrania uzla v topológii. Každé rozhranie má svoje meno, opis, má pridelenú IP adresu, podsieťovú masku, dĺžku prefixu, MAC adresu, jeho status, status protokolu, rýchlosť rozhrania a šírku pásma. Okrem týchto informácií sú tu ešte dodatočné dve dátové štruktúry, ktoré opisujú informáciu spojenú s prenosovými vlastnosťami rozhrania a suseda, t.j. uzol na druhej strane rozhrania. Prenosové

vlastnosti sú vyjadrené pomocou šírky pásma, oneskorenia, počtu paketov a bytov, ktoré boli spracované na vstupe a výstupe a taktiež počtu zaznamenaných chýb. Susedné zariadenie je definované pomocou jeho mena, IP adresy, MAC adresy a názvu rozhrania, cez ktoré je uzol prepojený so susedným rozhraním.

Výpis 6.4: Dátová štruktúra hrany

```

1 {
2   "edge_id": "id_of_edge",
3   "start_node_id": "id_of_one_node",
4   "start_node_interface": "name_of_one_interface",
5   "start_node_address": "ip_address_of_one_interface",
6   "end_node_id": "id_of_second_node",
7   "end_node_interface": "name_of_second_node",
8   "end_node_address": "ip_address_of_second_node",
9   "enabled": true,
10  "speed": "speed_of_link",
11  "maximum_bandwidth": "maximum_bandwidth_of_link",
12  "availableBandwidth": "available_bandwidth_of_link",
13  "existing_flows": []
14 }
```

Hrany sú definované pomocou dátovej štruktúry zobrazenej v kóde č. 6.4. Informácia, ktorá je v nej uchovaná, je identifikátor hrany, identifikátor uzla, rozhrania a IP adresy oboch uzlov, ktoré daná hrana prepája, či je daná linka zapnutá, rýchlosť prenosu cez danú linku, maximálna a dostupná šírka pásma a taktiež zoznam všetkých dátových tokov využívajúcich danú hranu.

Tieto dátové štruktúry sú v implementácii reprezentované pomocou tried. Tieto triedy uchovávajú informácie dané v dátových štruktúrach, ktoré boli opísané vyššie, pričom aj sprostredkujú interakciu medzi sieťovým modelom a prostredím. To je realizované tak, že počas modelovania správania počítačovej siete, ktorej implementácia bude opísaná v podsekcii č. 6.2, sa aktuálny stav počítačovej siete uchováva v inšanciách ich tried. Tieto objekty taktiež dovoľujú navrátiť stav počítačovej siete do pôvodného stavu. To je realizované tým, že pôvodný tvar dátových štruktúr, z ktorých sú vytvorené inšancie tried, je uložený v inštancii triedy reprezentujúcej topológiu ako celok. Keďže trieda reprezentujúca celú topológiu zodpovedá za vytvorenie inšancií všetkých potrebných tried, pri obnovení stavu počítačovej siete sa využijú pôvodné dáta pre znovuvytvorenie reprezentácie počítačovej siete. Okrem informácie z dátových štruktúr, tieto

objekty dovoľujú udržiavať dodatočnú informáciu o stave počítačovej siete, napríklad čiastočnú informáciu o predchádzajúcom stave počítačovej siete, pričom tieto informácie sa udržiavajú najmä pre zjednodušenie implementácie sieťového modelu.

Okrem tried reprezentujúcich dáta zo spomínaných dátových štruktúr, dodatočne je implementovaná trieda *Edge*, ktorej inštancie reprezentujú jednotlivé dátové toky. Tieto objekty v sebe udržiavajú informáciu o dátových tokoch, pričom najdôležitejšie informácie sú zdroj a cieľ dátového toku, jeho požiadavky, aj keď v rámci tejto práce všetky požiadavky všetkých dátových tokov sú jednotné, informáciu o stratovosti a oneskorení dátového toku. Okrem týchto informácií je v objektoch uchovávaná aj informácia o počte krokov, ktoré bude ešte daný dátový tok aktívny v počítačovej sieti.

6.2 Implementácia sieťového modelu

Na základe navrhovaného sieťového modelu z podsekcii č. 5.5.2, bola implementovaná statická funkcia *check_flows*, ktorá, na základe parametra obsahujúceho inštanciu triedy *Topology*, modeluje správanie počítačovej siete v jednom časovom kroku. Táto funkcia sa skladá z troch podstatných podkrokov, konkrétne *__generate_number_of_packets*, *__calculate_loss_delay_per_edge* a *__update_flows*.

Prvý podkrok slúži na náhodné vygenerovanie počtov paketov, ktoré v danom kroku majú byť smerované z počiatočného uzla dátového toku do jeho cieľového uzla. Ako bolo spomínané v podsekcii č. 5.5.2, táto práca predpokladá rovnaké počty paketov v oboch smeroch komunikácie.

V druhom kroku sa modeluje správanie samotnej počítačovej siete, pričom cieľom je modelovať očakávanú stratovosť paketov a očakávané oneskorenie paketov na každej linke. Taktiež sa tu modeluje, ktoré pakety z ktorých dátových tokov budú zahodené na každej linke medzi zdrojovým a cieľovým uzlom pre dané dátové toky.

V poslednom kroku sa aktualizujú hodnoty všetkých existujúcich dátových tokov. V tomto prípade to znamená, že sa aktualizuje hodnota *TTL*, ktorá hovorí koľko má daný dátový tok ešte byť aktívny, odhadované oneskorenie a stratovosť na základe liniek, cez ktoré je dátový tok smerovaný a podobne.

Nové dátové toky sa vytvárajú na požiadavku prostredia a za ktorú zodpovedá prostredie, ako je opísané v sekcii č. 6.1. Samotná funkcia *check_flows*, ktorá, ako bolo vyššie opísané, primárne zodpovedá za modelovanie práce počítačovej siete, je volaná podľa potreby prostredím, ktoré bolo opísané v sekcii č. 6.1.

6.3 Implementácia smerovacích prístupov

Navrhované smerovacie prístupy sú implementované v dvoch hlavných častiach, prvou časťou je implementácia agenta, ktorá zodpovedá za vykonanie smerovacích rozhodnutí, či už výber cesty alebo predikcia metriky, a druhá časť zabezpečuje interakciu medzi agentom, prostredím a čiastočne, v prípade tejto práce, modelom siete.

Pri tejto interakcii, ako bolo už aj napísané, agent je zodpovedný za smerovacie rozhodnutia, prostredie realizuje udržiavanie stavu siete, zabezpečuje získanie stavu siete a odmeny, vytvorenie nového dátového toku a priradenie nového dátového toku do reprezentácie počítačovej siete. Spomínaná druhá časť zabezpečuje vykonávanie týchto interakcií ako aj ich správne poradie. Pri spomínaných experimentoch, druhou časťou je trénovacia slučka vo fáze trénovania a testovacia slučka vo fáze evaluácie navrhovaných smerovacích prístupov. Obe slučky vo svojej základnej forme fungujú nasledovným spôsobom:

1. Zabezpečia vytvorenie nového dátového toku.
2. Zabezpečia existenciu reprezentácie stavu počítačovej siete pre agenta.
3. Zabezpečia vykonanie smerovacieho rozhodnutia agentom a, podľa potreby, upraví akciu do vhodného tvaru.
4. Zabezpečia vykonanie modelovania časového kroku počítačovej siete. V tomto kroku taktiež poskytne informáciu o novom dátovom toku a smerovacej politike s ním asociovannej.
5. Zabezpečia uchovanie potrebných dát pre neskoršie vyhodnotenie trénovania agenta, respektíve práce agenta.

Okrem týchto krokov je na začiatku niektorých epizód trénovania možné simulovať prácu siete, čo v tomto prípade znamená niekoľko krokov vytvárania smerovacích rozhodnutí bez pomoci agentov. Taktiež po ukončení vyššie spomínaných krokov, je potrebné zabezpečiť úpravu politík agentov na základe nazbieraných skúsenosti z už vykonaných krokov, to sa vykonáva až po dosiahnutí stanoveného počtu krokov, a periodické uloženie aktuálneho stavu agenta pre jeho možné neskoršie použitie.

Agenti su implementovaní podľa opisu prístupov strojového učenia v sekcii č. 5.4. V nasledujúcich podsekcích budú tieto prístupy strojového učenia spomenuté pre účely navrhovaných smerovacích prístupov.

6.3.1 Agent pre predikciu metriky

Agent pre predikciu metriky je implementovaný na základe prístupov strojového učenia opísaných v sekcii č. 5.4. Konkrétne pre agenta pre predikciu metriky je využitá metóda TD3 opísaná v podsekcii č. 5.4.1. Ako aj pri implementácii prostredia, reprezentácie stavu a odmeny, tak aj v tomto prípade je vytvorená všeobecná trieda, ktorá opisuje všetky vlastnosti, ktoré musí zohľadňovať implementácie agenta pre predikciu metriky. Táto trieda definuje nasledujúce metódy:

- *step* – je metóda, ktorá zabezpečuje vykonanie predikcií pre všetky hrany topológie, vyžaduje si dva parametre, a to *state* a *add_noise*, kde prvý parameter predstavuje stav siete a druhý hovorí, či sa má pridať šum k predikovaným akciám,
- *update_policy* – zabezpečuje úpravu parametrov neurónovej siete počas tréningu,
- *save* – slúži na uloženie aktuálneho stavu agenta pre jeho neskoršie použitie,
- *load* – slúži na načítanie uloženého stavu agenta z uloženého súboru,
- *get_optimizers* – slúži na získanie všetkých referencií na optimalizátory využívané neurónovou sieťou agenta.

Na základe tejto množiny potrebných metód je potom možné implementovať inštancie agenta pre predikciu metriky.

Konkrétne inštancie implementácie agenta pre predikciu metriky potom obsahujú všetko potrebné pre ich správnu prácu. Každá takáto inštancia aj dodatočne stanovuje aké parametre, respektíve hyperparametre, je potrebné stanoviť pre ich prácu. Konkrétne hyperparametre ako aj ich nastavenie sú opísané v kapitole č. 7. V tejto časti spomenieme dôležité vlastnosti implementácie agenta pre predikciu metriky bez ohľadu na hyperparametre tréningu. Pri inicializácii inštancie agenta, je potrebné zadať dimenzionalitu stavu a dimenzionalitu akcií. Taktiež je potrebné zadať typ buffer-a, ktorý má byť použitý pri tréningu pre ukladanie získaných skúseností. Ostatné parametre zodpovedajú hyperparametrom prístupu strojového učenia, ktorý implementuje daná inštancia agenta.

Pri inicializácii inštancie je taktiež nutné inicializovať neurónové siete, ktoré realizujú samotné predikcie. V tejto implementácii je to realizované nasledovným spôsobom; pre každú potrebnú neurónovú sieť je vytvorená inštancia modelu, ktorý zo seba predstavuje triedu, kde je uložené všetko potrebné pre prácu

neurónovej siete a taktiež predstavuje rozhranie medzi agentom a neurónovou sieťou, čo dovoľuje vyššiu abstrakciu implementácie.

6.3.2 Agent pre výber cesty

Podobne ako je implementovaný agent pre predikciu metriky, aj agent pre výber cesty implementuje prístupy strojového učenia opísané v sekcii č. 5.4. V tomto prípade sa ale jedná o prístupy realizujúce DQN, najdôležitejšia je jeho modifikácia Rainbow, ako je opísané v podsekcii č. 5.4.3, a prístupy realizujúce PPO, ako je opísané v podsekcii č. 5.4.2.

DQN

Aj v tomto prípade je vytvorená abstraktná trieda, tak ako pri agentovi pre predikciu metriky, kde sú ale v tomto prípade dodatočne aj implementované niektoré pomocné metódy, ktoré sú spoločné pre všetky implementácie. Táto trieda definuje nasledujúce metódy:

- *select_action* – je metóda, ktorá zabezpečuje vykonanie vybratia jednej možnej cesty pre smerovanie dát, na rozdiel od predikcií metriky ale využíva dve metódy pre výber jednej náhodnej cesty alebo najlepšej cesty na základe predikcie metriky,
- *update_policy* – zabezpečuje úpravu parametrov neurónovej siete počas trénovania,
- *save* – slúži na uloženie aktuálneho stavu agenta pre jeho neskoršie použitie,
- *load* – slúži na načítanie uloženého stavu agenta z uloženého súboru,
- *update_target_networks* – zabezpečuje úpravy cieľových sietí,
- *reduce_epsilon* – zabezpečuje postupné znižovanie parametra ϵ , ktorý určuje pomer náhodných akcií a najlepších akcií podľa agenta,
- *get_optimizers* – slúži na získanie všetkých referencií na optimalizátory využívané neurónovou sieťou agenta.

Okrem týchto metód, abstraktná implementácia má čiastočnú implementáciu pre uchovávanie potrebných spoločných parametrov medzi viacerými rôznymi implementáciami ako aj zabezpečuje vytvorenie buffer-a.

V tejto časti taktiež ešte budú spomenuté parametre inštancií, ktoré nezahŕňujú hyperparametre tréovania opísané v kapitole č. 7.1. Tieto parametre sú najmä nasledovné parametre; dimenzionality stavového priestoru a dimenzionality akčného priestoru, spolu s typom buffer-a sú spoločné parametre, ktoré majú aj inštancie spomínané v predchádzajúcej podsekcii venovanej agentom pre predikciu metriky, taktiež ale je tu potrebný parameter, ktorý určuje, či sa majú vôbec vykonávať náhodné akcie agentom a či je potrebné znižovať hodnotu parametra počas tréovania.

Aj v tomto prípade sa pri vytvorení inštancie agenta vytvárajú neurónové siete podobným spôsobom ako pri agentovi pre predikciu metriky. Aj tu je vytvorená abstrakcia medzi neurónovou sieťou zaobalenou do vlastnej triedy a samotným agentom.

PPO

Na rozdiel od predchádzajúcich implementácií, v tomto prípade nebola vytvorená abstraktná reprezentácia triedy agenta, keďže tu nie je predpoklad existencie viacerých implementácií tohto algoritmu, ako bolo pri agentovi pre predikciu metriky (predikcia jednej alebo viacerých metrík), a pri agentovi pre výber akcie s využitím DQN (viacero modifikácií pôvodného algoritmu DQN). Agent využívajúci algoritmus PPO je implementovaný v jednej triede. Táto trieda definuje nasledujúce metódy:

- *select_action* – je metóda, ktorá zabezpečuje vykonanie vybratia jednej možnej cesty pre smerovanie dát,
- *update_policy* – zabezpečuje úpravu parametrov neurónovej siete počas tréovania,
- *save* – slúži na uloženie aktuálneho stavu agenta pre jeho neskoršie použitie,
- *load* – slúži na načítanie uloženého stavu agenta z uloženého súboru,
- *update_target_networks* – zabezpečuje úpravy cieľových sietí,
- *get_optimizers* – slúži na získanie všetkých referencií na optimalizátory využívané neurónovou sieťou agenta.

V tejto časti taktiež ešte budú spomenuté parametre inštancií, ktoré nezahŕňujú hyperparametre tréovania opísané v kapitole č. 7.1. Tieto parametre sú najmä nasledovné parametre; dimenzionality stavového priestoru a dimenzionality

akčného priestoru, tieto parametre sú spoločné s parametrami ostatných implementácií agentov, koeficienty definujúce akým podielom majú vplývať na celkovú vypočítanú stratu podiely straty kritika a entropie a maximálne hodnoty do ktorých budú ohraňované pomery zmien akcií agentov oproti pôvodným akciám.

Aj v tomto prípade sa pri vytvorení inštancie agenta vytvárajú neurónové siete podobným spôsobom ako pri predchádzajúcich prístupoch.

6.3.3 Kombinácia oboch prístupov

Pre spoločnú prácu oboch agentov nie je potrebné nijakým spôsobom modifikovať samotnú implementáciu agentov ani modelu neurónovej siete. Je ale potrebné upraviť tréningovú a testovaciu slučku tak, aby z výstupu agenta pre predikciu metriky sa vytvoril vhodný stav pre agenta pre výber cesty. To znamená, že je potrebné modifikovať základnú formu pôvodných slučiek pre tréning a testovanie aby vyzerali nasledovne:

1. Zabezpečiť vytvorenie nového dátového toku.
2. Zabezpečiť existenciu reprezentácie stavu počítačovej siete pre agenta pre predikciu metriky.
3. Zabezpečiť vytvorenie nových predikcií metrík pre daný stav siete.
4. Zabezpečiť nájdenie n ciest medzi zdrojovým a cieľovým uzlom.
5. Zabezpečiť vykonanie smerovacieho rozhodnutia agentom pre výber cesty a, podľa potreby, upraviť akciu do vhodného tvaru.
6. Zabezpečiť vykonanie modelovania časového kroku počítačovej siete. V tomto kroku taktiež poskytnúť informáciu o novom dátovom toku a smerovacej politike s ním asociovanou.
7. Zabezpečiť uchovanie potrebných dát pre neskoršie vyhodnotenie tréningu agenta, respektíve práce agenta.

Samozrejme, rovnaké doplnenia, ako boli opísané na začiatku sekcie, ktoré sú potrebné pre tréning samotného agenta, musia byť zohľadnené aj v tomto prípade. Okrem týchto zmien nie sú vyžadované žiadne dopĺňujúce zmeny pre správnu prácu agentov spoločne.

7 Priebeh experimentov

Táto kapitola je venovaná opisu experimentov, ktorých cieľom je získať výsledky, na základe ktorých bude možné porovnať navrhnuté prístupy, opísané v sekciách č. 5.3 a č. 6.3, medzi sebou, ako aj ich porovnať s bežnými smerovacími protokolmi, ako je OSPF a RIP. Kapitola je rozdelená na dve sekcie. Prvá sekcia č. 7.1 opisuje priebeh experimentov, to jest, dobu ich trvania, porovnávané vlastnosti, nastavenia topológie a podobne. Druhá sekcia č. 7.2 opisuje špecifický priebeh tréningu agentov, ktorí sú využívaní ako reprezentanti navrhovaných prístupov.

7.1 Experimenty

Experimenty boli vykonávané na sieťovom modeli, tak ako je opísaný v podsekcii č. 5.5, a ktorý bol implementovaný tak, ako je opísané v podsekcii č. 6.2. V experimentoch je využitých 5 topológií, všetky sú opísané čiastočne v podsekcii č. 5.5.6, z opísaných topológií pre účely experimentov nie je využitá veľmi malá topológia (10Node). V experimentoch sa používajú dve hodnoty K_m , konkrétne 10 v prípade liniek s nižšou priemernou priepustnosťou a 100 s vyššou priepustnosťou liniek. Doplňujúce informácie k jednotlivým topológiám je možné vidieť v tabuľke č. 7.1. V tabuľke je možné vidieť vlastnosti topológií z pohľadu počtu hrán a liniek, ktoré ich tvoria, ale aj nastavenia použité pri tréningu a experimentoch. Tieto hodnoty definujú počty paketov, ktoré sa generujú pre jednotlivé dátové toky a nastavenie maximálneho počtu krokov, ktoré môže dátový tok byť aktívny v topológii.

Ako bolo opísané v sekcii č. 5.5.2, pre modelovanie aktivity počítačovej siete je v každom časovom kroku pridaný nový dátový tok do počítačovej siete. Tieto dátové toky sú vytvárané pomocou generácie čísel pseudo-náhodným generátorom čísel, ktorý sa využíva na vygenerovanie vlastností komunikácie ako aj na výber uzlov, ktoré sú pre daný dátový tok zdrojové a cieľové. Priebeh vyzerá nasledovne:

Tabuľka 7.1: Sumár vlastností a nastavení topológií

Topológia	Grid25	SwitchBlock	GEANT4	20Nodes	30Nodes
Uzly	25	10	47	20	30
Hrany	40	19	65	27	42
Životnosti tokov	300 – 500	200 – 300	300 – 500	300 – 500	300 – 500
Počet paketov	10 – 20	10 – 20	10 – 20	10 – 20	10 – 20

1. Najprv sa pseudo-náhodne vyberie zdrojový uzol v_i zo zoznamu všetkých uzlov topológie.
2. Pseudo-náhodne sa vyberie cieľový uzol v_j taktiež zo zoznamu všetkých uzlov topológie.
3. Nakoniec sa náhodne pridelia všetky parametre smerovania daného dátového toku.
4. Pre tento dátový tok je nájdená cesta p_{ij} , pričom sa použije smerovací prístup, ktorý sa využíva v danom experimente.
5. Tento nový dátový tok sa pridá do zoznamu všetkých aktívnych dátových tokov.
6. V každom kroku sú skontrolované všetky aktívne dátové kroky, pričom sa dekrementuje ich hodnota životnosti TTL . Pri tomto a nasledujúcom kroku sa vynecháva najnovší dátový tok.
7. Všetky dátové toky, ktorých životnosť klesla na hodnotu 1, sú odstránené zo zoznamu aktívnych dátových tokov.
8. Následne sa modeluje správanie siete s aktívnymi dátovými tokmi. Pri tom sa využíva náhodná hodnota počtu paketov pre každý aktívny dátový krok v danom časovom kroku. Maximálny a minimálny počet týchto paketov je daný rozsahom stanoveným pre každú topológiu.

Tieto kroky sa vykonávajú v každom kroku experimentu.

Pre porovnanie výsledkov dosiahnutých navrhovanými prístupmi v porovnaní s bežnými smerovacími protokolmi, boli vykonané experimenty s použitím navrhovaných smerovacích prístupov a bežných smerovacích prístupov na topológiách opísaných v predchádzajúcich paragrafoch. Výsledky jednotlivých

prístupov boli primárne porovnávané na základe priemernej stratovosti l_t , ktorá bola kalkulovaná pre každý aktívny dátový tok v čase t nasledovne:

$$l_t = \frac{1}{|F_t|} \sum_{i=0}^{|F_t|} l_{t_i}, \quad (7.1)$$

kde i je index existujúceho aktívneho dátového toku v čase t , a l_{t_i} je odhadovaná stratovosť pre i -ty dátový tok v čase t a $|F_t|$ počet všetkých aktívnych dátových tokov v čase t .

Okrem stratovosti boli použité aj iné metriky pre porovnanie výsledkov rôznych smerovacích prístupov. Špecificky to bol priemerný čas odozvy d_t vypočítaný pre každý dátový tok v čase t ,

$$d_t = \frac{1}{|F_t|} \sum_{i=0}^{|F_t|} d_{t_i}, \quad (7.2)$$

kde i je index aktívneho dátového toku v čase t , $|F_t|$ je počet všetkých aktívnych dátových tokov v čase t , a d_{t_i} priemerné oneskorenie i -teho dátového toku v čase t . Ďalšou metriku je počet prázdnych, nevyužitých liniek e_{empty_t} v čase t , ako aj priemerný počet dátových tokov asociovaných s linkou f_t v čase t , a celková priemerná stratovosť pre dátové linky l dosiahnutá za celý priebeh experimentu,

$$l = \frac{1}{n_{\text{steps}}} \sum_{t=0}^{n_{\text{steps}}} l_t, \quad (7.3)$$

kde n_{steps} je počet krokov experimentu.

Každý experiment pozostáva z 10,000 modelovaných časových krokov. Keďže stabilný stav počítačovej siete bol dosahovaný až pri 600-tom až 800-tom časovom kroku, konkrétna hodnota závisí od použitej topológie, prvých 1,000 časových krokov bolo zahodených a nepoužitých z pohľadu vypočítaných štatistík a vykreslených grafov. Konkrétne, 1,000 časových krokov bolo zvolených pre zahodenie z dôvodu, že v tom kroku už boli zaručene všetky modelované siete v stálom stave, bez ohľadu na použitú topológiu v experimente, a zároveň poskytuje aj určitú rezervu pre prípad, že počítačová sieť potrebuje ešte o niekoľko viac krokov pre stabilizáciu. Ostatné dáta boli využité pre vytvorenie štatistík a nakreslenie grafov, ktoré budú zobrazené v kapitole č. 8. Keďže pre účely experimentu sa dátové toky a ich vlastnosti generujú náhodne, pre zabezpečenie rovnakých dátových tokov v každom experimente je nastavený seed knižníc použitých pre generovanie týchto hodnôt na rovnaké hodnoty. Výsledky dosiahnuté v experimentoch sú použité na porovnanie všetkých navrhnutých prístupov pre

smerovanie medzi sebou, ako aj pre porovnanie ich s výsledkami dosiahnutými smerovacími prístupmi, ktoré využívajú algoritmus SPF pre nájdenie najkratšej cesty, pričom pre modelovanie správania OSPF protokolu sú využité váhy liniek, ktoré boli dopočítané tak, ako sú vypočítané pri použití protokolu OSPF, a pre modelovanie správania RIP sú využité váhy liniek, ktoré boli dopočítané tak, ako sú vypočítané pri použití protokolu RIP.

Experimenty boli vykonané na počítači vybavenom procesorom Intel Core I7-13700H CPU @ 3.7 GHz a s 16GB primárnej pamäte.

7.2 Trénovanie

V tejto sekcii bude opísaný proces trénovania agentov pre výber cesty a predikciu metriky. Hyperparametre agentov použité pri experimentoch boli pri každom experimente rovnaké. Hodnoty hyperparametrov je možné vidieť v tabuľkách č. 7.2, č. 7.4 a č. 7.5, ktoré obsahujú hyperparametre použité pre trénovanie oboch agentov a pri využití rozdielnych algoritmov pre trénovanie agenta pre výber cesty. Trénovanie bolo separované do 10,000 časových krokov dlhých epizód, pričom každá epizóda začína z čistého, inak povedané, prázdneho stavu modelu počítačovej siete z pohľadu aktívnych dátových tokov. Prvá polovica epizód realizovala zahrievaciu fázu. Táto fáza trvá 500 krokov, pričom v týchto sa modelovalo správanie siete bez účasti agenta, aby sme dosiahli stály stav siete pred jeho zapojením do smerovacieho procesu. Druhá polovica epizód už túto fázu nemá.

Tabuľka 7.2: Hyperparametre použité pre trénovanie agenta pre predikciu metriky

Parametre	Hodnoty
Rýchlosť učenia kritika	$1 * 10^{-4}$
Rýchlosť učenia aktéra	$1 * 10^{-3}$
Optimalizátor	AdamW
τ	$5 * 10^{-3}$
γ	0.95
Veľkosť dávky	256

Agent pre predikciu metriky mal nastavenú hodnotu rýchlosti učenia na $1 * 10^{-4}$ v prípade kritika a na hodnotu $1 * 10^{-3}$ v prípade aktéra, AdamW bol použitý pre aktualizáciu hodnôt váh neurónových sietí, parameter τ bol nastavený na

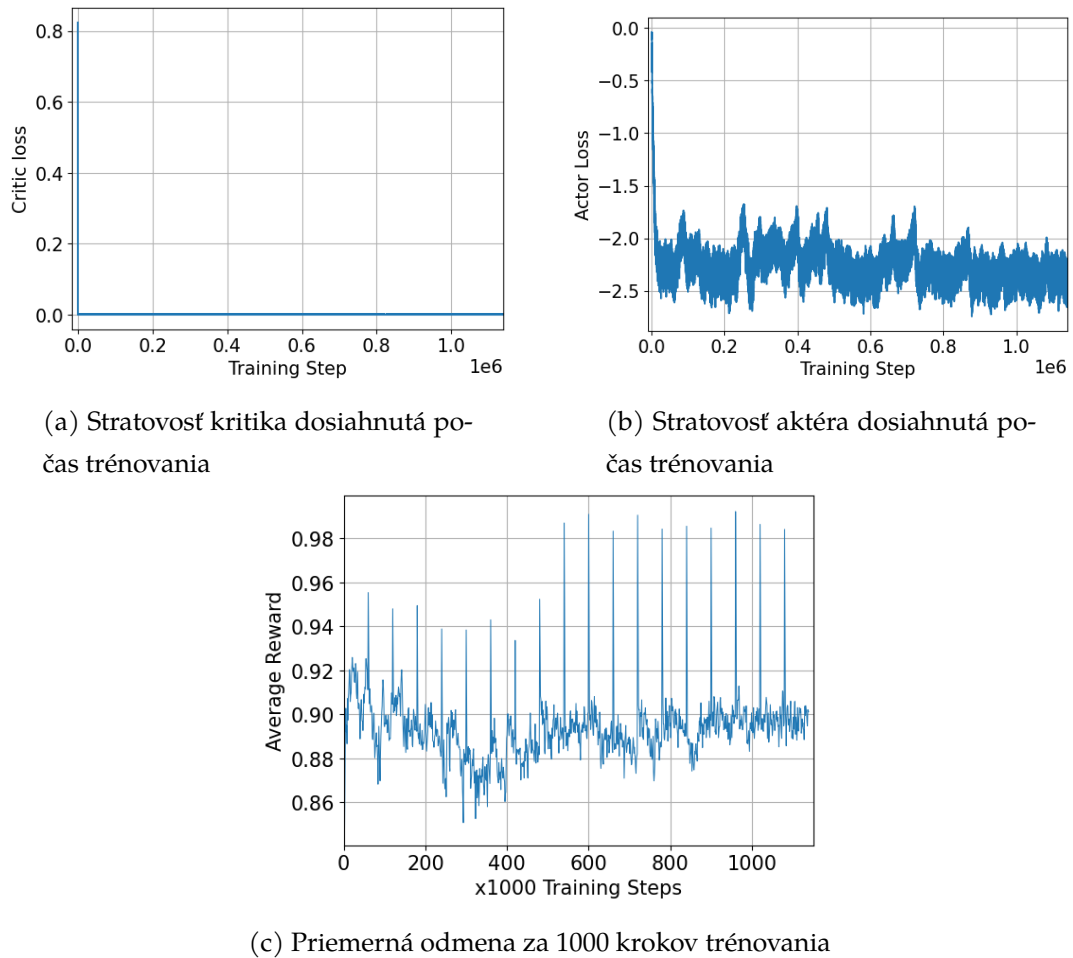
hodnotu $5 * 10^{-3}$, parameter γ mal hodnotu 0.95, a veľkosť dávky bola nastavená na 256, taktiež boli cieľové siete aktéra a kritika upravované každé dva kroky. Prioritizovaný buffer [45] bol použitý pre uchovávanie a získavanie skúseností. Parameter α bol nastavený na hodnotu 0.6 a β na hodnotu 0.4. Počas tréovania bol využívaný Ornstein-Uhlenbeck šum [46] a bol aplikovaný na predikované hodnoty akcie neurónovou sieťou. Parametre šumu je možné vidieť v tabuľke č. 7.3. Parameter μ bol nastavený na hodnotu 0.0, parameter θ na hodnotu 0.15, parameter σ na hodnotu 0.05, parameter $\sigma_{\min}$ na hodnotu 0.015 a parameter σ_{decay} na hodnotu 0.975. Neurónové siete aktéra, kritikov a ich cieľových sietí boli vytvorené pomocou štyroch vrstiev lineárnych neurónových sietí. Prvé tri vrstvy všetkých týchto sietí majú rovnakú štruktúru; prvá, vstupná, vrstva má tvar poskytnutého stavu, druhá a tretia vrstva sú tvorené 400 a 200 neurónmi, po každej vrstve je aplikovaná funkcia Rectified Linear Unit (ReLU) na ich výstup. Neurónové siete kritikov majú výstupnú vrstvu tvorenú jedným neurónom, ako aj neurónové siete aktérov, keďže taký je tvar akčného priestoru, ale na výstup aktérov je ešte aplikovaná funkcia tanh. Tréovanie trvalo 1.2 milióna krokov, čo korešponduje s 120 epizódami, v prípade samostatnej predikcie váhy liniek. Keďže pri súčasnej predikcii všetkých liniek nie je tréovaný agent genericky na viacerých topológiách, bolo tréovanie o niečo kratšie, konkrétne len 1 milión krokov.

Tabuľka 7.3: Parametre Ornstein-Uhlenbeck šumu

Parameter	Hodnota
μ	0.0
θ	0.15
σ	0.05
σ_{decay}	0.975
$\sigma_{\min}$	0.015

Jeden agent pre samostatnú predikciu metriky jednej linky bol natréovaný. Tento agent bol následne použitý pri všetkých vykonaných experimentoch, v ktorých vystupoval agent pre predikciu metriky pri ich samostatnej predikcii. Obrázky č. 7.1a a č. 7.1b zobrazujú stratu kritika a aktéra pri tréovaní. Ako je možné vidieť, strata kritika rýchlo upadla na začiatku tréovania. Strata aktéra sa stabilizovala neskôr, až okolo 860,000-tého tréovacieho kroku. Na obrázku č. 7.1c je zobrazená priemerná odmena za každých 1000 krokov, ktorú dostal agent za svoje akcie počas tréovania.

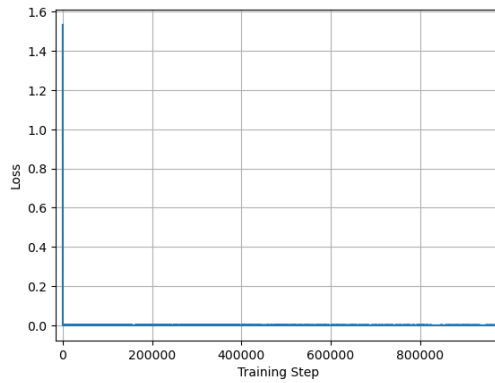
Príklad výsledkov dosiahnutých pri tréovaní agenta pre predikciu metriky



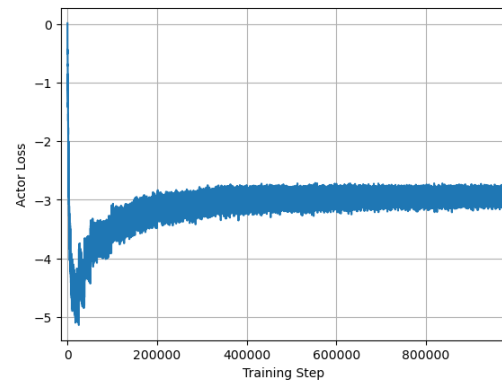
Obrázok 7.1: Výsledky tréningu dosiahnuté agentom pre samostatnú predikciu metriky počas tréningu

pri súčasnej predikcii všetkých váh liniek je možné vidieť na obrázku č. 7.2a a č. 7.2b, z pohľadu straty, a na obrázku č. 7.2c je možné vidieť dosiahnutú odmenu za každých 1000 krokov tréningu.

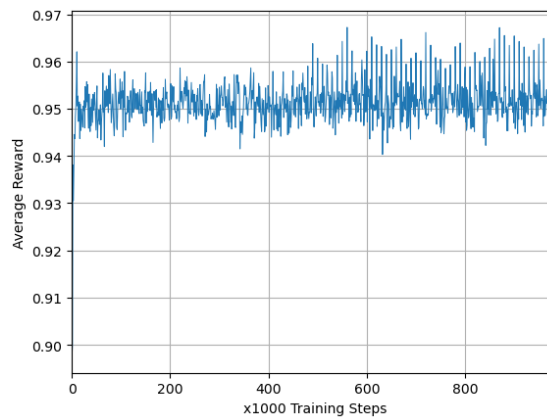
Agent pre výber cesty s využitím algoritmu DQN rovnako používa optimalizátor AdamW. Rýchlosť učenia bola nastavená na hodnotu $1 * 10^{-4}$, parameter γ bol nastavený na hodnotu 0.99, parameter multistep bol nastavený na 10, počet quantilov bolo nastavených na 20, τ mal nastavenú hodnotu 0.05. Taktiež bol využívaný prioritizovaný buffer s rovnakými parametrami, konkrétne parameter α bol nastavený na hodnotu 0.6 a parameter β na hodnotu 0.4. Neurónová sieť používaná agentom obsahuje šumivé neurónové vrstvy [44]. Parameter determinujúci počiatočné správanie Gaussovskej štandardnej deviácie bol nastavený na hodnotu 0.01. Agent bol tréňovaný $2 * 10^6$ krokov, čo korešponduje s 200 epizódami. Neurónová sieť agenta je formovaná z niekoľkých vrstiev, vstupná vrstva má tvar reprezentácie stavu, dve lineárne vrstvy o 256 neurónoch, následne je



(a) Stratovosť kritika dosiahnutá počas trénovania



(b) Stratovosť aktéra dosiahnutá počas trénovania



(c) Priemerná odmena za 1000 krokov trénovania

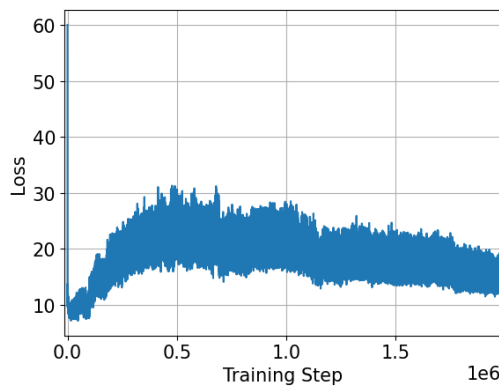
Obrázok 7.2: Výsledky trénovania dosiahnuté agentom pre súčasnú predikciu metriky počas trénovania

šumivá vrstva, ktorá predikuje priemerné hodnoty quantilov pre daný stav, táto vrstva má 256 neurónov a druhá šumivá vrstva, ktorá má tvar počtu quantilov, a dve šumivé vrstvy, kde prvá má 256 neurónov a druhá má tvar výstupu, čo je súčin počtu quantilov a tvaru akčného prostredia. Medzi každou vrstvou bola aplikovaná na ich výstup ReLU funkcia. Konečný výstup je suma hodnôt quantilov pre stav a pre každú akciu samostatne. V každom kroku trénovania môže agent zvoliť cestu z 10 možných ciest. Táto sada ciest môže byť získaná pomocou SPF algoritmu so statickými váhami liniek alebo s dynamickými váhami liniek získanými pomocou agenta pre predikciu metriky.

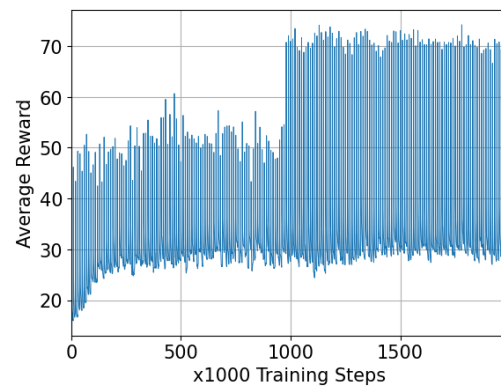
Na obrázku č. 7.3a je zobrazená strata agenta pre výber cesty s využitím DQN pri trénovaní. Ako je možné vidieť, na začiatku sa strata rýchlo znížila, ale neskôr sa zvyšovala do asi 400,000-teho kroku. Následne sa pomaly znižovala počas trénovacieho procesu. Na obrázku č. 7.3b je zobrazená priemerná odmena agenta pri trénovaní za 1000 krokov. Ako môžeme vidieť, do 500,000-ceho kroku sa od-

Tabuľka 7.4: Hyperparametre použité pri tréňovaní agenta pre výber cesty s využitím DQN

Parameter	Hodnota
Rýchlosť učenia	$1 * 10^{-4}$
γ	0.99
Multistep	2
Počet quantilov	20
τ	0.05



(a) Strata dosiahnutá agentom pre výber cesty počas tréňovania



(b) Priemerná odmena za tisíc krokov pri tréňovaní

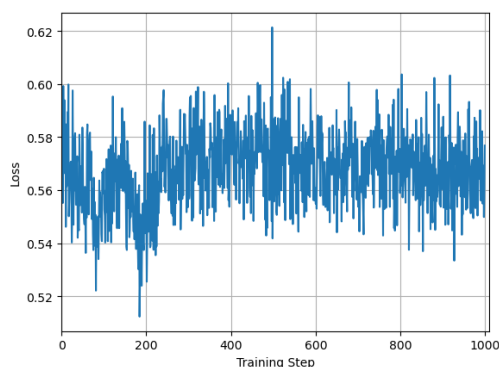
Obrázok 7.3: Dosiahnuté výsledky pri tréňovaní agenta pre výber cesty s využitím DQN

mena pomaly zvyšovala, kde sa aj nakoniec stabilizovala. Neskôr sa maximálna dosiahnutá hodnota odmeny zvýšila okolo $1 * 10^6$ -tého kroku.

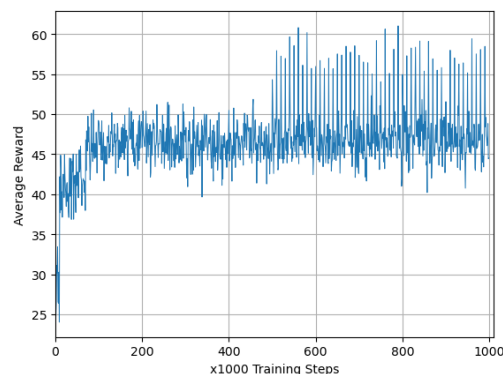
Tabuľka 7.5: Hyperparametre použité pri tréňovaní agent pre výber cesty s využitím algoritmu PPO

Parameter	Hodnota
Rýchlosť učenia	$1 * 10^{-3}$
ϵ	0.2
γ	0.95
Koeficient entropie	0.1
Veľkosť dávky	1000

Agent pre výber cesty s využitím algoritmu PPO taktiež používa optimalizátor AdamW. V tomto prípade je nastavená rýchlosť učenia na hodnotu $1 * 10^{-3}$, hodnota parametra γ je nastavená na 0.95, parametra ϵ na 0.2, koeficient entropie



(a) Strata dosiahnutá agentom pre výber cesty počas tréovania



(b) Priemerná odmena za tisíc krokov pri tréovaní

Obrázok 7.4: Dosiahnuté výsledky pri tréovaní agenta pre výber cesty s využitím PPO

pie bol nastavený na hodnotu 0.1, tréovanie agenta bolo vykonané každých 1000 krokov, to je, keď sa nahromadilo nových 1000 skúseností v buffer-i a tréovanie na každej dávke skúseností bolo vykonané 10 krát. V tomto prípade sa nevyužíva prioritizovaný buffer na rozdiel od predchádzajúcich implementácií. Neurónová sieť je tvorená vstupnou vrstvou o veľkosti reprezentácie stavu, dvoma vrstvami o 256 neurónoch. Ich výstup je vstupom pre kritiku, ktorý je zložený z jednej vrstvy s 256 neurónami a výstupnej vrstvy, ktorá vracia jednu hodnotu, a je vstupom aj pre aktéra, ktorý je taktiež zložený z jednej vrstvy s 256 neurónami, ale jeho výstup je tvorený vektorom, ktorého dimenzionalita závisí na počte možných akcií.

Na obrázkoch č. 7.4a a č. 7.4b je možné vidieť priebeh tréovania agenta pre výber cesty s využitím algoritmu PPO. Ako je možné vidieť, strata agenta neklesala výrazne počas tréovania. Na začiatku tréovania mala najprv tendenciu klesať, ale po 190000 krokoch tréovania znova začala stúpať, pričom okolo 290000 kroku sa stabilizovala a až do konca tréovania ostala približne na rovnakej hodnote. Na druhej strane, odmena dosahovaná agentom sa najprv zvýšila do 150000 kroku, kde sa stabilizovala. Výkyvy zaznamenané po 500000 krokoch sú spôsobené tým, že po polovici tréovania už agent vyberá cesty od začiatku epizódy, to je, z čistého stavu počítačovej siete.

8 Vyhodnotenie

Táto kapitola je venovaná opisu výsledkov experimentov opísaných v kapitole č. 7. Porovnania výsledkov dosiahnutých rôznymi smerovacími prístupmi sú vykonané na základe stratovosti, oneskorenia, počtu dátových tokov na linkách a počtu prázdnych liniek.

Na základe experimentov boli pre experimenty zvolené nasledovné reprezentácie stavov:

- Pre samostatnú predikciu váh liniek boli zvolené nasledujúce parametre pre reprezentáciu stavu jednej linky, podiel stále dostupnej šírky pásma oproti toeretickej maximálnej šírky pásma, maximálna šírka pásma škálovaná do hodnôt medzi $[0, 1]$, očakávané oneskorenie škálované medzi $[0, 1]$ a podiel prichádzich paketov k maximálnemu počtu paketov, ktoré je možné na linke spracovať.
- Pre spoločnú predikciu váh liniek boli zvolené nasledujúce parametre rovnako ako pri samostatnej predikcii váh liniek, pričom sa takto získané reprezentácie liniek spojili za sebou podľa indexov liniek v reprezentácii topológií do jedného stavového vektora.
- Pre spoločnú predikciu váh liniek bola taktiež odskúšaná varianta reprezentácie, kde boli využité rovnaké informácie ako pri predchádzajúcich možnostiach, pričom bola vytvorená matica susednosti v tvare $n \times n \times d$, kde n je počet vrcholov topológie a d je počet parametrov, v našom prípade je d rovné 4.
- Pre výber cesty bola využitá reprezentácia stavu v tvare $n \times d$, kde n je počet možných ciest, v našom prípade je zvolená hodnota $n = 10$, a d je počet sledovaných parametrov, v našom prípade je $d = 15$. Stav bol reprezentovaný pomocou priemerných, minimálnych a maximálnych hodnôt dosiahnutých na hranách každej linky, ktorá by bola využívaná i -tou cestou. Sledované hodnoty boli stratovosť paketov, oneskorenie, dostupná šírka pásma, počet tokov a podiel paketov k maximálne spracovateľným paketom na linke.

Prvá reprezentácia stavu je využívaná agentom pre predikciu metriky, pričom v prípade, že je predikovaná každá váha linky samostatne. Druhá a tretia reprezentácia je využívaná pre predikovanie liniek váh v prípade, ak sa ich všetky snažíme predikovať v jednom kroku. Posledná reprezentácia je využívaná pri výbere cesty, a to aj agentom využívajúcim PPO aj DQN.

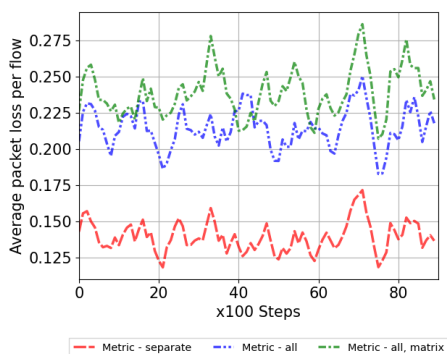
Táto kapitola je rozdelená na tri sekcie. Sekcia č. 8.1 sa venuje porovnaniu výsledkov dosahovaných samostatnými agentmi s rôznou reprezentáciou stavov. Sekcia č. 8.2 sa venuje porovnaniu výsledkov najlepších samostatných agentov s kombinovaným prístupom a prístupmi využívajúcimi algoritmus SPF. Posledná sekcia č. 8.3 je venovaná krátkemu zhrnutiu dosiahnutých výsledkov.

8.1 Dosiahnuté výsledky samostatných prístupov

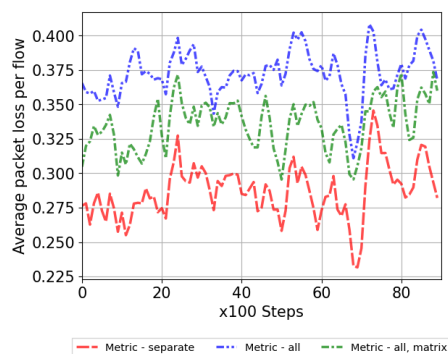
Najprv budú ukázané porovnania výsledkov dosiahnutých agentom pre predikciu metriky v závislosti od predikcie váh liniek súčasne a samostatne, pričom v prípade súčasnej predikcie všetkých liniek váh je vykonané porovnanie aj na základe využitej reprezentácie stavu. V tomto porovnaní, ako aj pri ostatných porovnovaniach, bol využitý agent pre samostatnú predikciu váh liniek natrénovaný na všetkých topológiách súčasne, ostatní agenti boli trénovaní špecificky pre každú topológiu.

Na obrázku č. 8.1 je možné vidieť, aké boli pri experimentoch dosiahnuté priemerné stratovosti paketov jednotlivých prístupov k predikcii váh liniek. Všeobecne najlepšie výsledky boli dosiahnuté pri využití samostatnej predikcie váh liniek oproti ich spoločnej predikcii. Jedinou výnimkou je topológia 30Nodes, kde mali oveľa lepšie výsledky agenti, ktorí predikovali všetky váhy liniek súčasne. Pri porovnaní rôznych reprezentácií stavov pre predikciu váh liniek súčasne nie je možné s istotou povedať, ktorá reprezentácia poskytuje lepšie výsledky. Pri niektorých topológiách (Grid25, GEANT4, 30Nodes) má lepšie výsledky druhá reprezentácia (nie matica susednosti), pri ostatných (SwitchBlock, 20Nodes) má lepšie výsledky tretia reprezentácia stavu (matica susednosti).

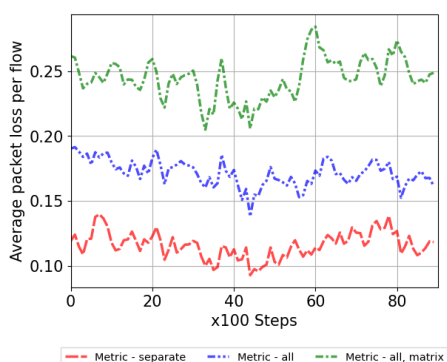
Na obrázku č. 8.2 je možné vidieť výsledky z pohľadu predikcie dosiahnutého oneskorenia. Aj v tomto prípade je možné povedať, že celkovo najlepšie výsledky dosahuje agent pri predikcii samostatne každej metriky, aj keď je tu výnimka vo forme topológie 30Nodes. Taktiež, nie je možné určiť, ktorá reprezentácia stavu je v tomto prípade lepšia pre súčasnú predikciu váh liniek. Na niektorých topológiách majú veľmi podobné výsledky, (Grid25, SwitchBlock), na dvoch (30Nodes a 20Nodes) je lepšia druhá reprezentácia stavu (nie matica



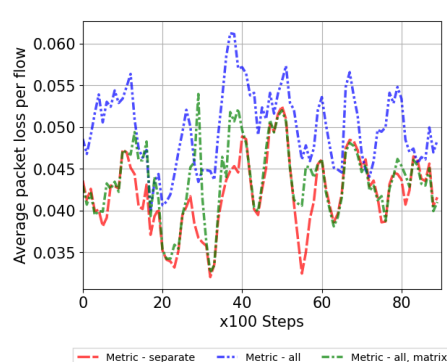
(a) Strata paketov na Grid25



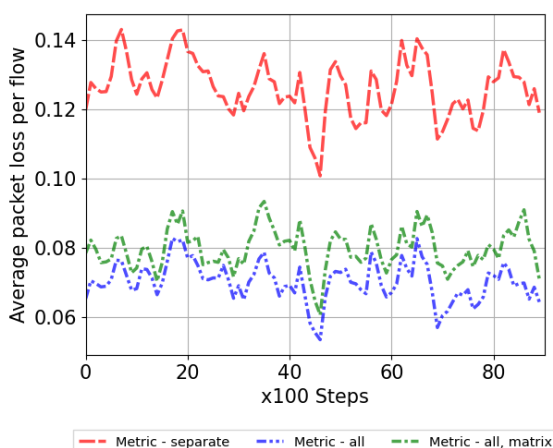
(b) Strata paketov na SwitchBlock



(c) Strata paketov na GEANT4



(d) Strata paketov na 20Nodes

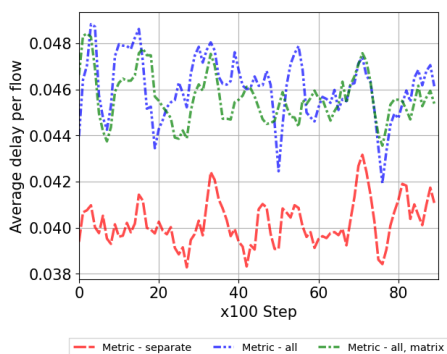


(e) Strata paketov na 30Nodes

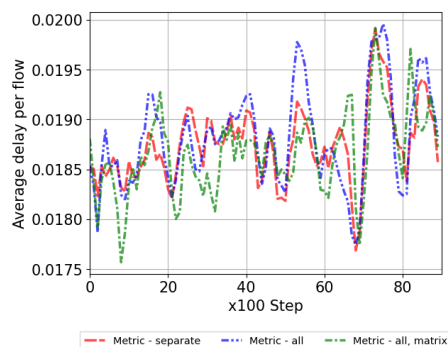
Obrázok 8.1: Strata paketov dosiahnutá pri experimentoch pri využití rôznych prístupov k predikcii váh liniek.

susednosti), na poslednej (GEANT4) je najlepšia tretia reprezentácia stavu siete (matica susednosti).

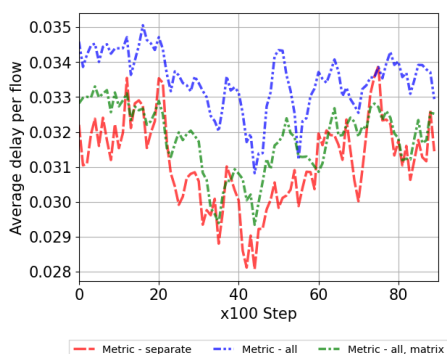
Na obrázku č. 8.3 je možné vidieť dosiahnuté výsledky z pohľadu počtu prázdnych (nevyužitých) liniek v počítačovej sieti. V tomto prípade má taktiež najlepšie výsledky samostatná predikcia váh liniek, pričom len pri topológii SwitchBlock sa dá povedať, že by mohli výsledky tohto agenta nebyť najlepšie. Ostatné prístupy využívajú menej liniek, inak povedané, nechávajú viac liniek prázdnych.



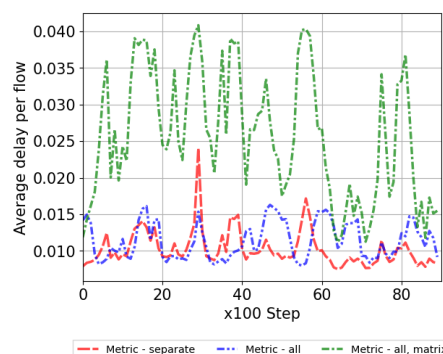
(a) Oneskorenie v milisekundách na Grid25



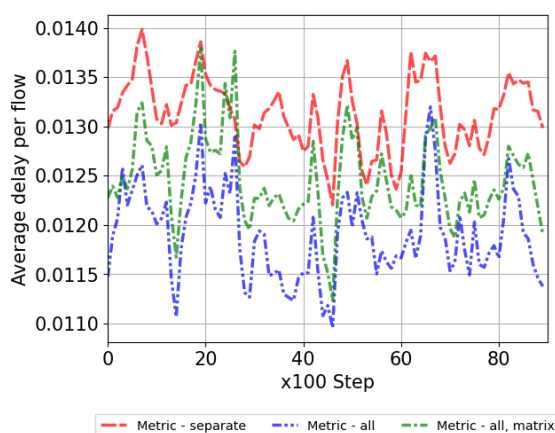
(b) Oneskorenie v milisekundách na SwitchBlock



(c) Oneskorenie v milisekundách na GEANT4



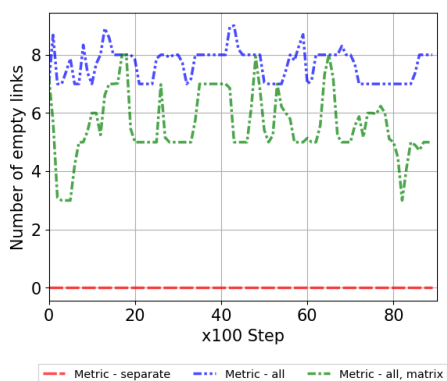
(d) Oneskorenie v milisekundách na 20Nodes



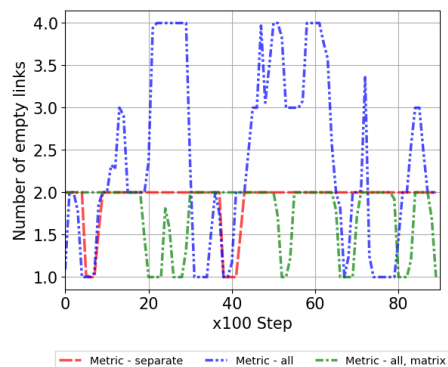
(e) Oneskorenie v milisekundách na 30Nodes

Obrázok 8.2: Oneskorenie v milisekundách dosiahnuté pri experimentoch pri využití rôznych prístupov k predikcii váh liniek.

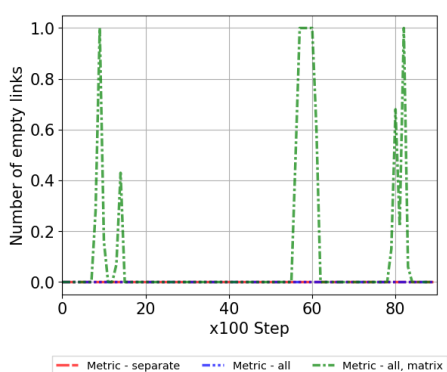
Aj v tomto prípade závisí od topológie to, ktorá reprezentácia stavu dosahuje lepšie výsledky pri predikcii všetkých váh liniek súčasne. Na dvoch topológiách (GEANT4 a 20Nodes) je to druhá reprezentácia stavu (nie matica susednosti) a na ostatných (30Nodes, Grid25, SwitchBlock) je to tretia reprezentácia stavu siete (matica susednosti).



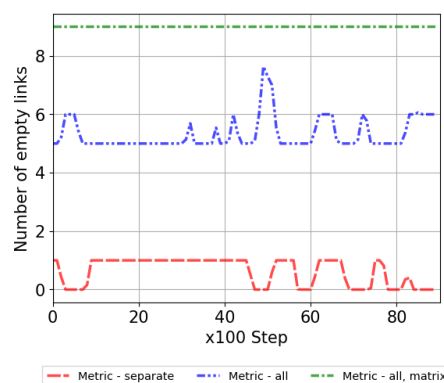
(a) Priemerné počty prázdnych liniek na Grid25



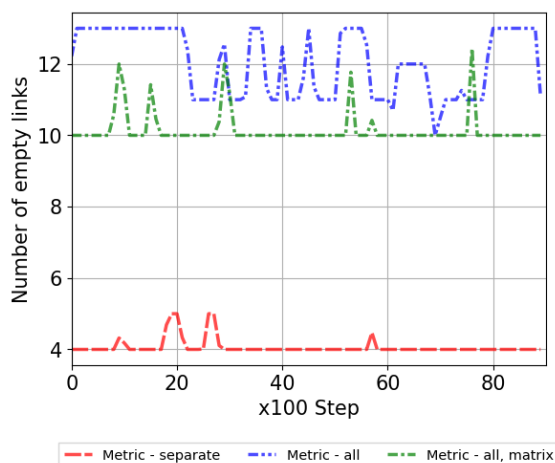
(b) Priemerné počty prázdnych liniek na SwitchBlock



(c) Priemerné počty prázdnych liniek na GEANT4



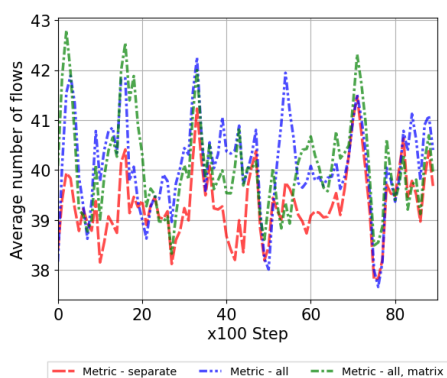
(d) Priemerné počty prázdnych liniek na 20Nodes



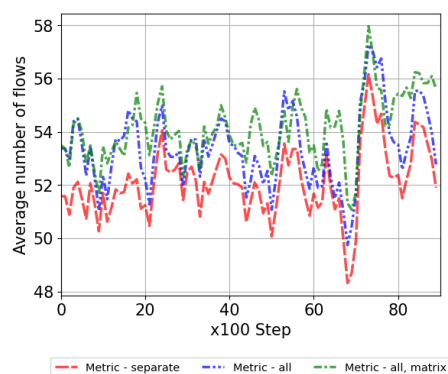
(e) Priemerné počty prázdnych liniek na 30Nodes

Obrázok 8.3: Priemerné počty prázdnych liniek pri experimentoch pri využití rôznych prístupov k predikcii váh liniek.

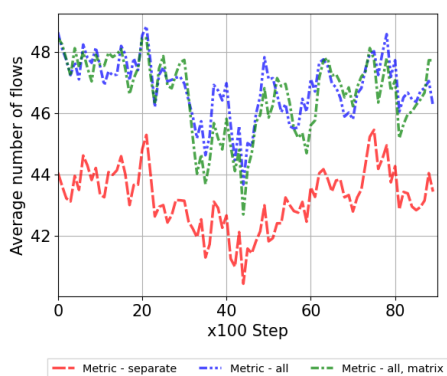
Na obrázku č. 8.4 sú zobrazené výsledky agentov dosiahnuté z pohľadu priemerného počtu dátových tokov prechádzajúcich cez linky počítačovej siete. V tomto prípade, najmenšie počty dátových tokov boli väčšinou dosiahnuté pri samostatnej predikcii váh liniek, okrem prípadu topológie (20Nodes). Pri ostatných



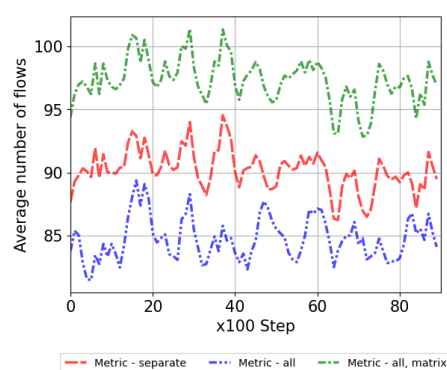
(a) Priemerné počty dátových tokov na linkách na Grid25



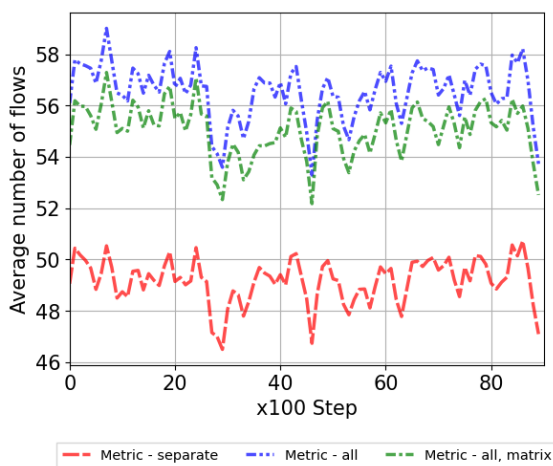
(b) Priemerné počty dátových tokov na linkách na SwitchBlock



(c) Priemerné počty dátových tokov na linkách na GEANT4



(d) Priemerné počty dátových tokov na linkách na 20Nodes



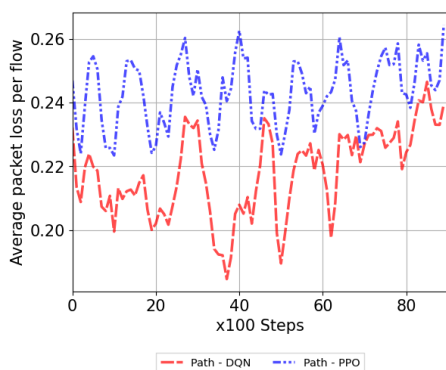
(e) Priemerné počty dátových tokov na linkách na 30Nodes

Obrázok 8.4: Priemerné počty tokov dát na linkách pri experimentoch pri využití rôznych prístupov k predikcii váh liniek.

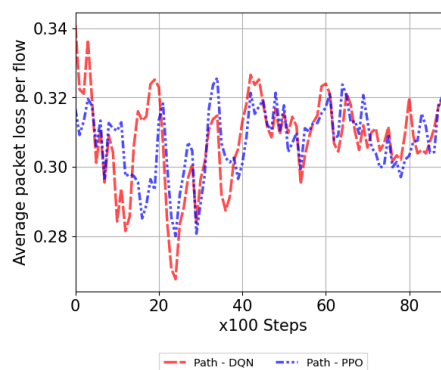
agentoch boli väčšinou výsledky veľmi podobné.

Nasledujúce porovnania sú vykonané medzi agentom pre výber cesty, ktorý je trénovaný algoritmom PPO a agentom trénovaným algoritmom DQN.

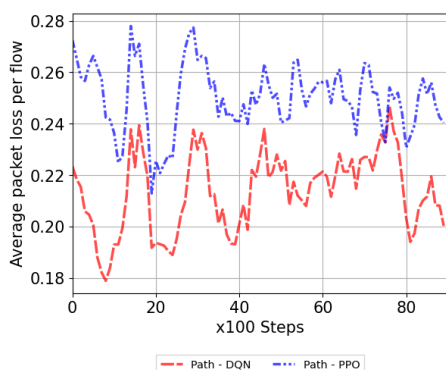
Na obrázku č. 8.5 je možné vidieť porovnanie agentov využívajúcich PPO



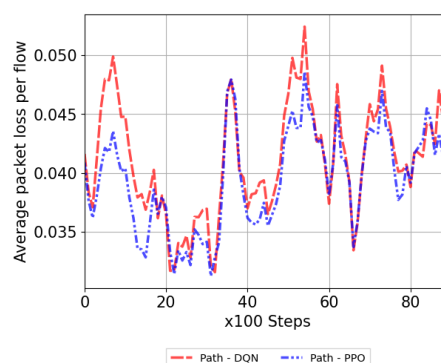
(a) Strata paketov na Grid25



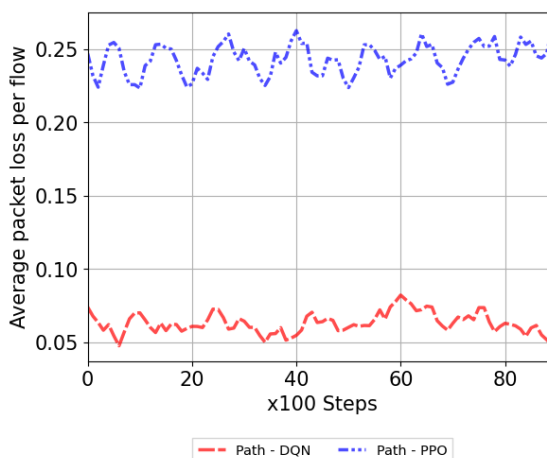
(b) Strata paketov na SwitchBlock



(c) Strata paketov na GEANT4



(d) Strata paketov na 20Nodes

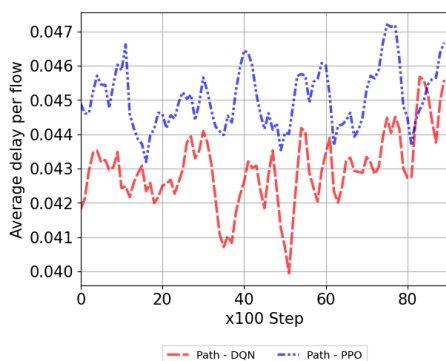


(e) Strata paketov na 30Nodes

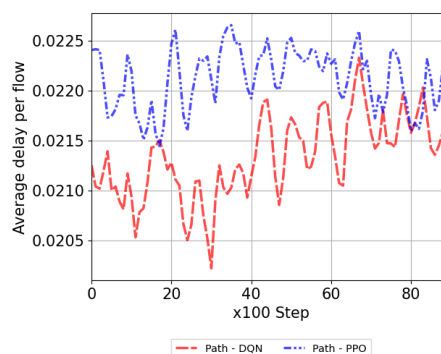
Obrázok 8.5: Strata paketov dosiahnutá pri experimentoch pri využití rôznych prístupov k výberu cesty.

a DQN na základe dosiahnutých hodnôt stratovosti paketov. Ako je možné vidieť, najlepšie výsledky všeobecne dosahuje agent s využitím algoritmu DQN. Takýto agent dosahoval lepšie výsledky na topológiách Grid25, GEANT4, 30Nodes. S ním porovnateľné výsledky dosahoval agent využívajúci PPO v prípade topológií SwitchBlock a 20Nodes.

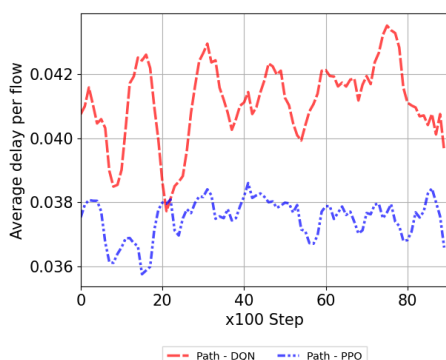
Na obrázku č. 8.6 sú vyobrazené výsledky porovnania agenta pre výber cesty



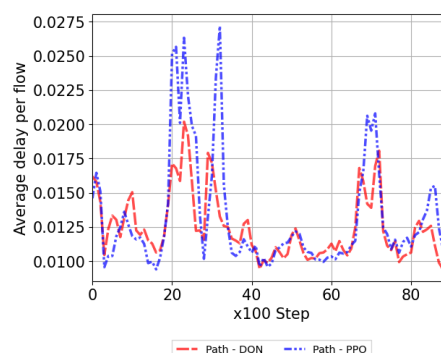
(a) Oneskorenie v milisekundách na Grid25



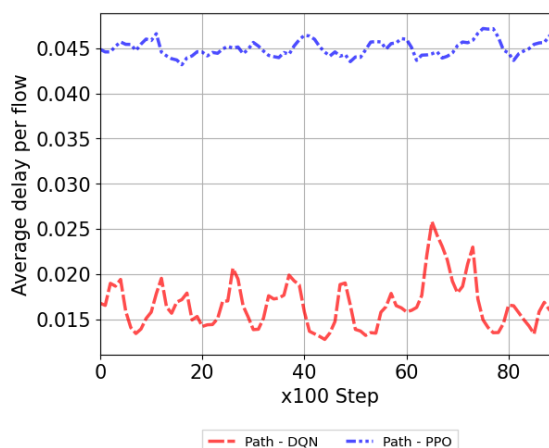
(b) Oneskorenie v milisekundách na SwitchBlock



(c) Oneskorenie v milisekundách na GEANT4



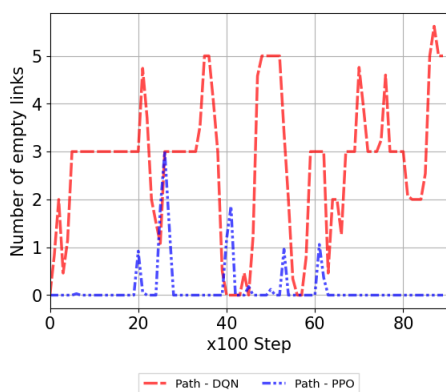
(d) Oneskorenie v milisekundách na 20Nodes



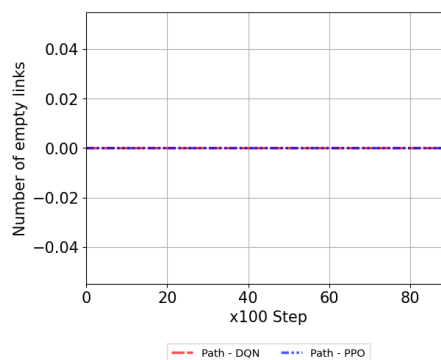
(e) Oneskorenie v milisekundách na 30Nodes

Obrázok 8.6: Oneskorenie v milisekundách dosiahnuté pri experimentoch pri využití rôznych prístupov k výberu cesty.

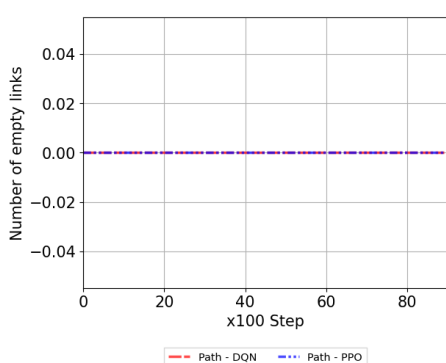
využívajúceho PPO a agenta využívajúceho DQN na základe oneskorenia paketov. Aj v tomto prípade dosahuje všeobecne najlepšie výsledky agent využívajúci pri trénovaní algoritmus DQN, len v jednom prípade mal agent využívajúci PPO lepšie výsledky, konkrétne pri topológii GEANT4. Taktiež pri topológii 20Nodes dosahoval agent využívajúci PPO porovnateľné výsledky s agentom využívajú-



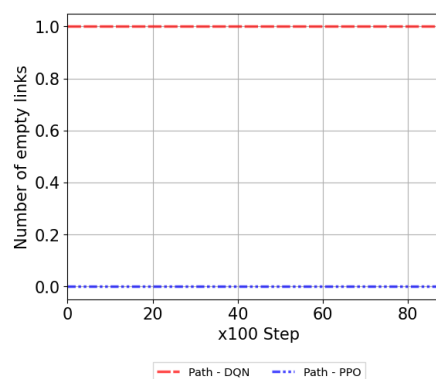
(a) Priemerné počty prázdnych liniek na Grid25



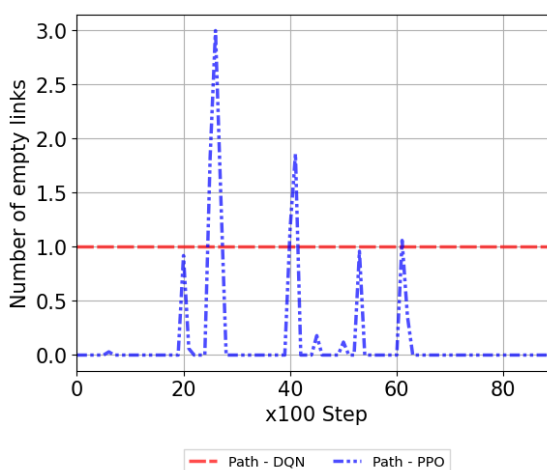
(b) Priemerné počty prázdnych liniek na SwitchBlock



(c) Priemerné počty prázdnych liniek na GEANT4



(d) Priemerné počty prázdnych liniek na 20Nodes

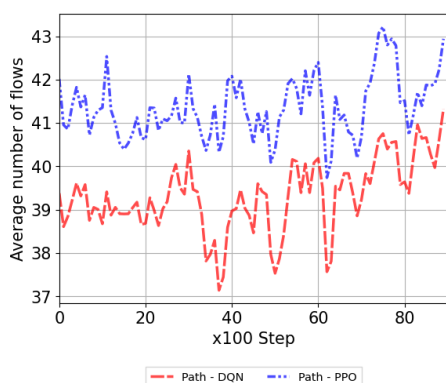


(e) Priemerné počty prázdnych liniek na 30Nodes

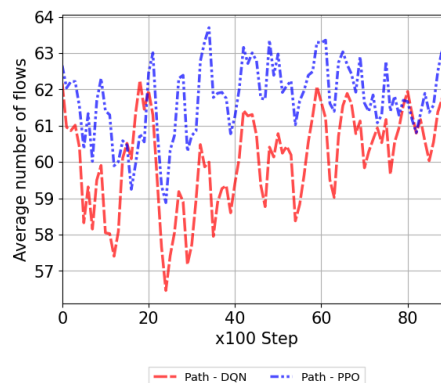
Obrázok 8.7: Priemerné počty prázdnych liniek pri experimentoch pri využití rôznych prístupov k výberu cesty.

cim DQN.

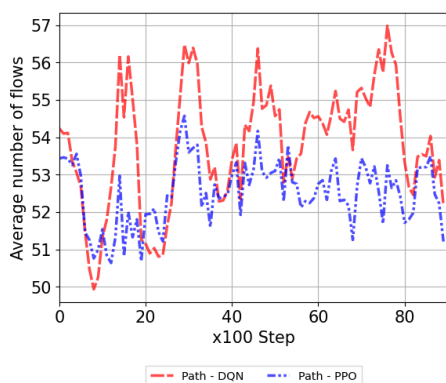
Na obrázku č. 8.7 je možné vidieť výsledky agenta pre výber cesty trénovaného algoritmom PPO a agenta trénovaného algoritmom DQN z pohľadu počtu



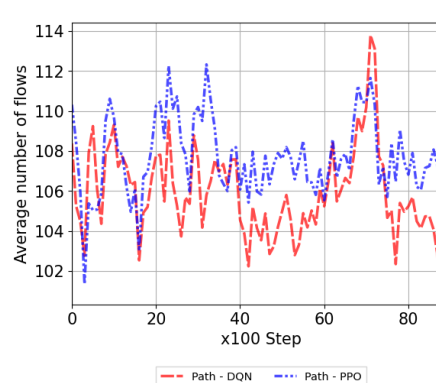
(a) Priemerné počty dátových tokov na linkách na Grid25



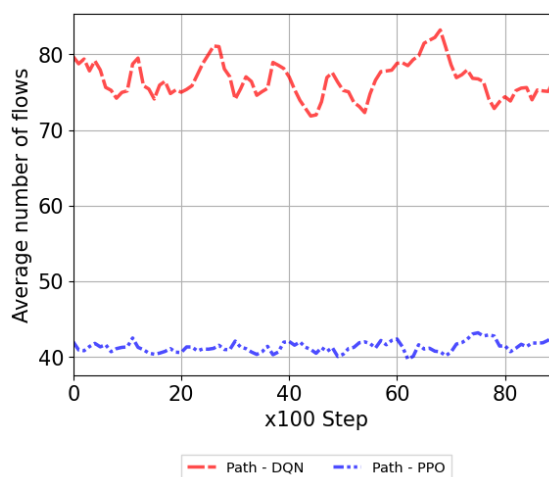
(b) Priemerné počty dátových tokov na linkách na SwitchBlock



(c) Priemerné počty dátových tokov na linkách na GEANT4



(d) Priemerné počty dátových tokov na linkách na 20Nodes



(e) Priemerné počty dátových tokov na linkách na 30Nodes

Obrázok 8.8: Priemerné počty tokov dát na linkách pri experimentoch pri využití rôznych prístupov k výberu cesty.

prázdnych liniek. V tomto prípade má všeobecne lepšie výsledky agent, ktorý pri tréňovaní využíval algoritmus PPO, pričom dosahoval lepšie výsledky v prípade troch topológií (Grid25, 20Nodes a 30Nodes). V ostatných prípadoch oba agenti

dosahovali rovnaké výsledky.

Na obrázku č. 8.8 sú zobrazené výsledky agenta pre výber cesty trénovaného algoritmom PPO a agenta trénovaného algoritmom DQN z pohľadu priemerného počtu dátových tokov na linkách počítačovej siete. Vo väčšine prípadov sú dosiahnuté výsledky veľmi podobné, pričom v troch prípadoch má lepšie výsledky agent využívajúci DQN (Grid25, SwitchBlock, 20Nodes) a v dvoch prípadoch má lepšie výsledky agent využívajúci PPO (GEANT4, 30Nodes).

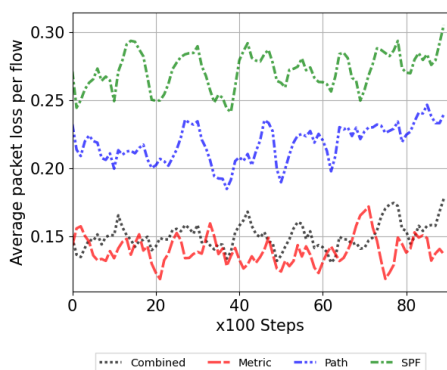
Na základe vyššie ukázaných výsledkov boli pre nasledujúce experimenty zvolené najlepšie prístupy. Rozhodovanie bolo vykonané najmä na základe dosiahnutých stratovostí agentov, pričom sa prihliadalo aj na ostatné sledované parametre spolu s vlastnosťami jednotlivých prístupov, ako je ich možnosť generického využitia na viaceré topológie počítačových sietí bez nutnosti dodatočného trénovania a upravovania ich architektúry. Takže pre ďalšie experimenty bola zvolená samostatná predikcia metriky a v prípade agenta pre výber cesty bol zvolený algoritmus DQN. Tieto implementácie boli následne využité pre ich kombinované využitie.

8.2 Dosiahnuté výsledky najlepších prístupov

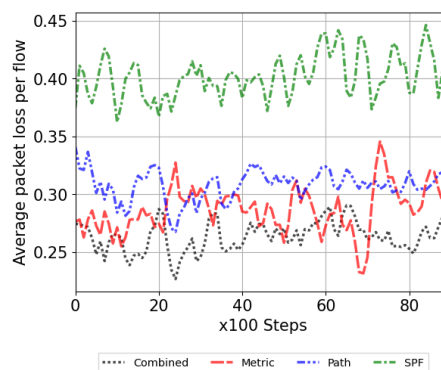
Tabuľka 8.1: Priemerná strata paketov dosiahnutá pri experimentoch smerovacím prístupmi podľa topológie.

Topology	Priemerná strata paketov			
	Kombinovaný	Výber cesty	Predikcia metriky	Najlepší SPF
Grid25	0.15	0.218	0.14	0.27
SwitchBlock	0.263	0.308	0.287	0.40
GEANT4	0.12	0.213	0.116	0.417
20Nodes	0.037	0.041	0.042	0.047
30Nodes	0.066	0.063	0.126	0.07

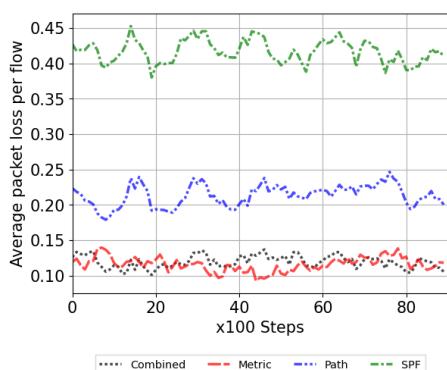
Na obrázku č. 8.9 je zobrazený vzťah medzi krokom experimentu a priemernou stratovosťou dosiahnutou smerovacími prístupmi. Priemerné hodnoty dosiahnuté pre kombinácie topológií a smerovacích prístupov sú zobrazené v tabuľke č. 8.1. Podľa použitej topológie pri experimente, vo väčšine prípadov, je zlepšenie výsledkov dosiahnutých smerovacími prístupmi využívajúcimi strojové učenie v porovnaní s prístupmi využívajúcimi výlučne SPF algoritmy sa pohybuje v rozsahu 8 percent až k zlepšeniu o približne 70 percent. Vo väčšine prí-



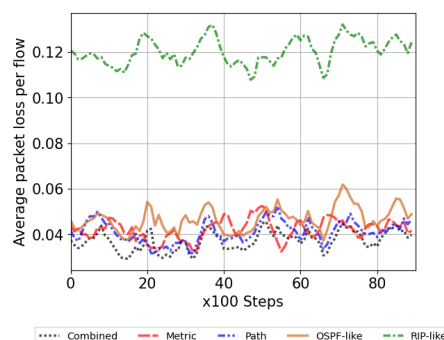
(a) Strata paketov na Grid25



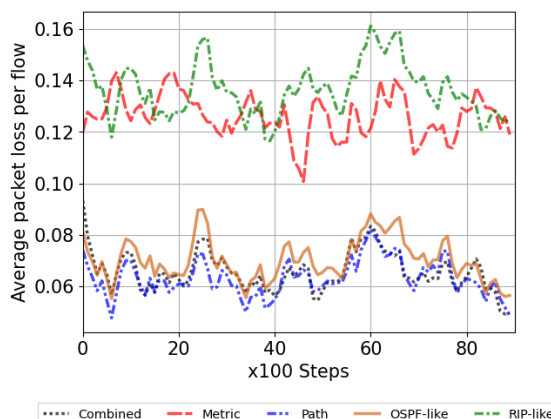
(b) Strata paketov na SwitchBlock



(c) Strata paketov na GEANT4



(d) Strata paketov na 20Nodes

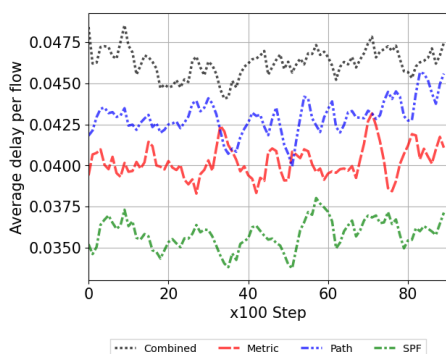


(e) Strata paketov na 30Nodes

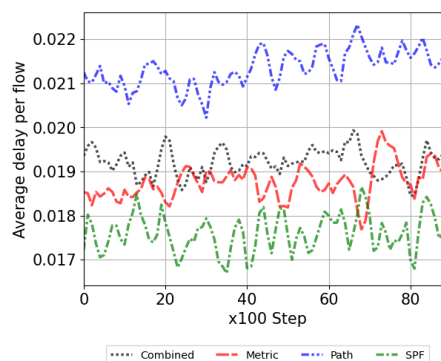
Obrázok 8.9: Strata paketov dosiahnutá pri experimentoch pri využití rôznych prístupov k smerovaniu.

padov boli najlepšie výsledky dosiahnuté použitím agenta pre predikciu metriky alebo oboma agentami spolu. Výnimkou ale je prípad experimentov na topológii 30Node. V tomto experimente sú výsledky dosiahnuté agentom pre predikciu metriky horšie ako pri využití lepšej metriky s SPF algoritmom. Na tejto topológii najlepšie výsledky dosahuje agent pre výber cesty.

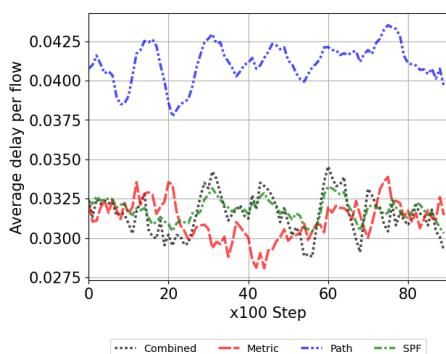
Na obrázku č. 8.10 môžeme vidieť výsledky dosiahnuté agentami z pohľadu priemerného oneskorenia dátových tokov. Pre lepšiu čitateľnosť výsledkov, nie-



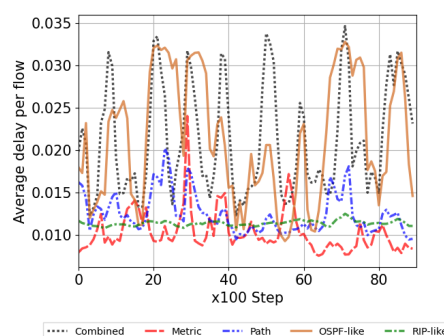
(a) Oneskorenie v milisekundách na Grid25



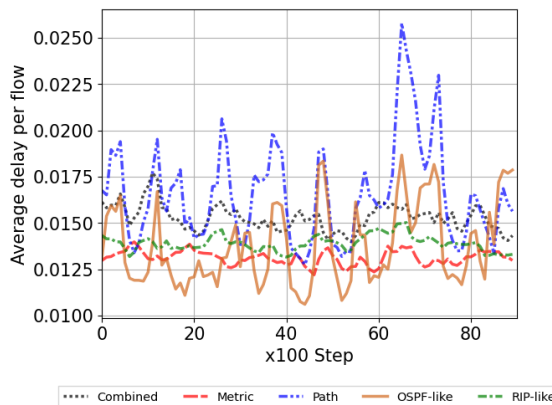
(b) Oneskorenie v milisekundách na SwitchBlock



(c) Oneskorenie v milisekundách na GEANT4



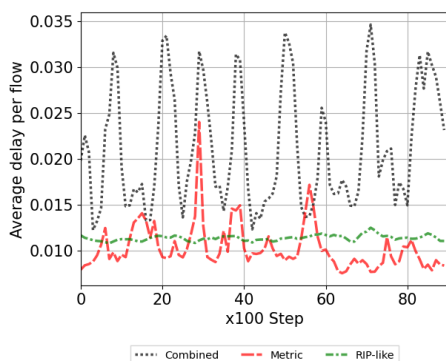
(d) Oneskorenie v milisekundách na 20Nodes



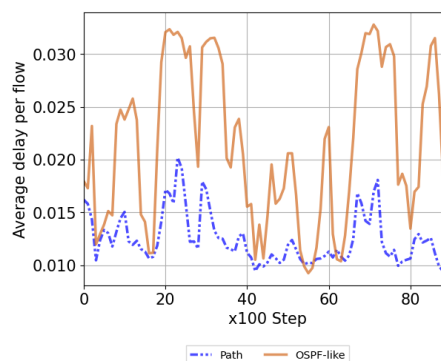
(e) Oneskorenie v milisekundách na 30Nodes

Obrázok 8.10: Oneskorenie v milisekundách dosiahnuté pri experimentoch pri využití rôznych prístupov k smerovaniu.

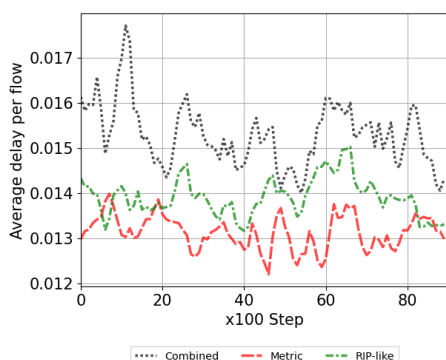
ktoré výsledky sú dodatočne zobrazené separátne na obrázku č. 8.11. Taktiež sú tieto dáta zapísané v tabuľke č. 8.2. Ako je možné vidieť, všeobecne sa oneskorenie zvýšilo s použitím navrhovaných smerovacích prístupov. Toto zvýšenie bolo nenápadné až veľmi vysoké. Najväčšie zvýšenie oneskorenia je až 33 percent. Iba v prípade jednej topológie, 20Node, bolo zaznamenané zníženie oneskorenia. Ale aj v tomto prípade došlo k zníženiu oneskorenia len v prípadoch oddeleného



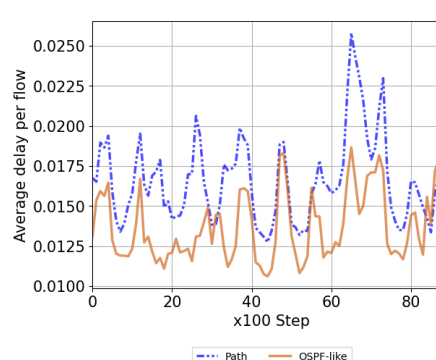
(a) Prvá polovica oneskorení na 20Node



(b) Prvá polovica oneskorení na 20Node



(c) Prvá polovica oneskorení na 30Node



(d) Prvá polovica oneskorení na 30Node

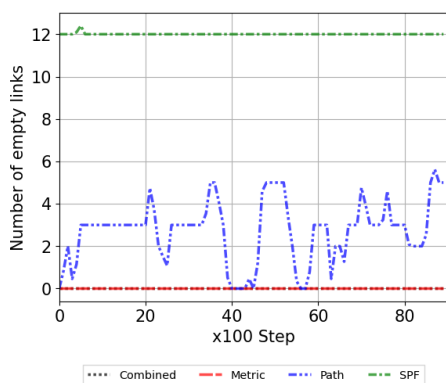
Obrázok 8.11: Strata paketov dosiahnutá pri experimentoch pri využití rôznych prístupov k smerovaniu, separátne dáta pre vybrané topológie.

Tabuľka 8.2: Priemerné oneskorenie v milisekundách dosiahnuté pri experimentoch smerovacím prístupmi podľa topológie.

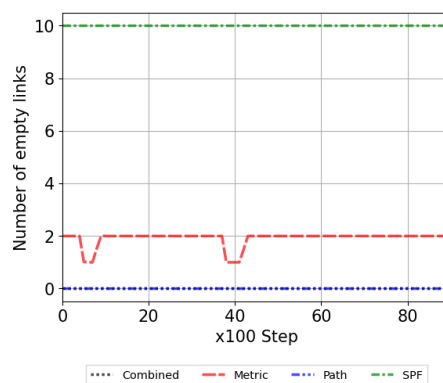
Topology	Priemerné oneskorenie			
	Kombinovaný	Výber cesty	Predikcia metriky	Najlepší SPF
Grid25	46.263	42.953	40.217	35.839
SwitchBlock	19.207	21.329	18.749	17.601
GEANT4	31.519	41.12	31.114	31.778
20Nodes	21.204	12.482	10.284	21.318
30Nodes	15.29	16.572	13.13	13.771

použitia agentov. Keďže oneskorenie v použitom smerovacom modeli neberie do úvahy oneskorenie spojené s potrebou znovu poslať pakety, ktoré boli zahodené, výber kratších ciest môže byť výhodnejší ako zníženie stratovosti využitím dlhších ciest.

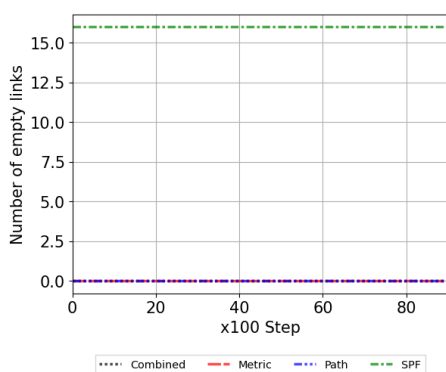
Na obrázku č. 8.12 je zobrazený priemerný počet prázdnych liniek. Tieto dáta



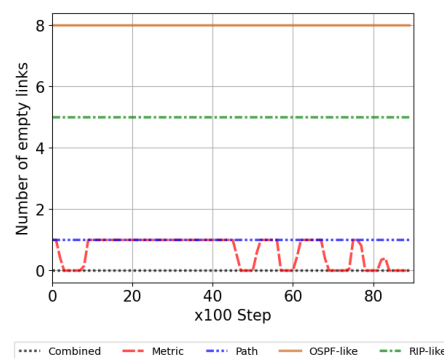
(a) Priemerné počty prázdnych liniek na Grid25



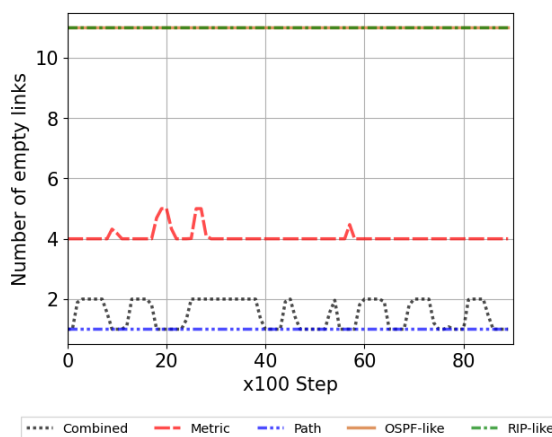
(b) Priemerné počty prázdnych liniek na SwitchBlock



(c) Priemerné počty prázdnych liniek na GEANT4



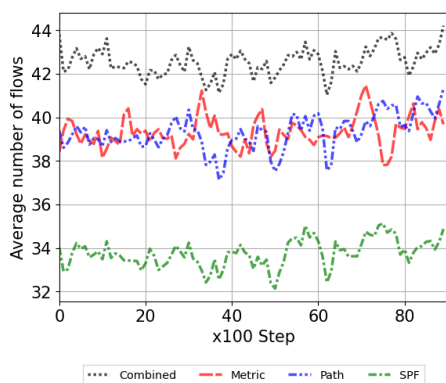
(d) Priemerné počty prázdnych liniek na 20Nodes



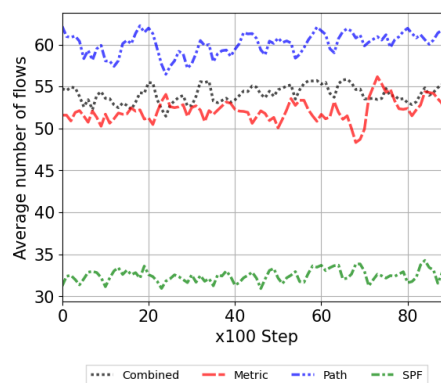
(e) Priemerné počty prázdnych liniek na 30Nodes

Obrázok 8.12: Priemerné počty prázdnych liniek pri experimentoch pri využití rôznych prístupov k smerovaniu.

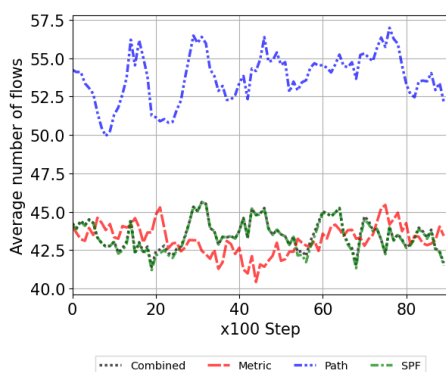
sú taktiež sumarizované v tabuľke č. 8.4. Ako je možné vidieť, najlepšie výsledky boli väčšinou dosiahnuté pri využití agenta pre výber cesty a pri spoločnom použití oboch prístupov. Najlepší stav, keď sú využité všetky linky, je zaznamenaný hodnotou 0. Všeobecne boli horšie výsledky dosiahnuté pri využití SPF



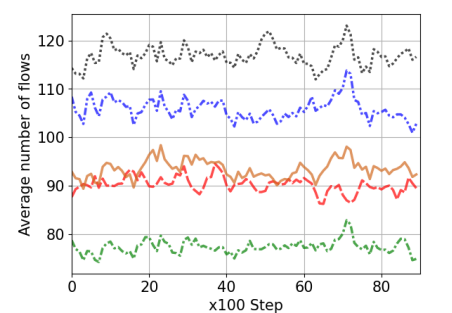
(a) Priemerné počty dátových tokov na linkách na Grid25



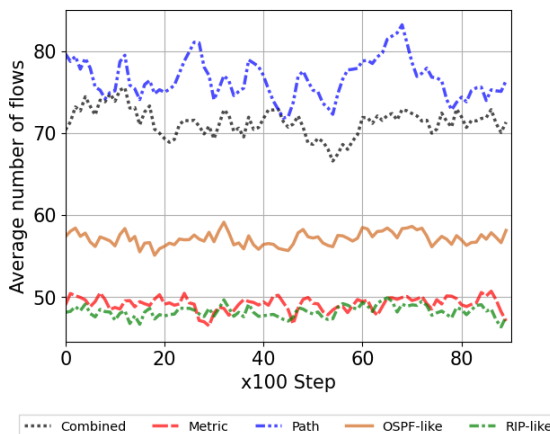
(b) Priemerné počty dátových tokov na linkách na SwitchBlock



(c) Priemerné počty dátových tokov na linkách na GEANT4



(d) Priemerné počty dátových tokov na linkách na 20Nodes



(e) Priemerné počty dátových tokov na linkách na 30Nodes

Obrázok 8.13: Priemerné počty tokov dát na linkách pri experimentoch pri využití rôznych prístupov k smerovaniu.

algoritmu. Tento prístup zanecháva až viac ako 10 prázdnych liniek.

Na obrázku č. 8.13 je zobrazený priemerný počet dátových tokov prechádzajúcich cez linku. Sumarizované výsledky je taktiež možné vidieť v tabuľke č. 8.3. Z týchto výsledkov vieme povedať, že v priemere je viac dátových tokov na linke

Tabuľka 8.3: Priemerný počet dátových tokov na linke dosiahnutých pri experimentoch smerovacím prístupmi podľa topológie.

Topology	Priemerný počet dátových tokov			
	Kombinovaný	Výber cesty	Predikcia metriky	Najlepší SPF
Grid25	42.557	39.312	39.321	33.761
SwitchBlock	54.077	60.084	52.106	32.447
GEANT4	43.508	53.787	43.205	43.427
20Nodes	116.887	105.909	90.193	93.434
30Nodes	71.356	76.606	49.174	57.135

Tabuľka 8.4: Priemerný počet prázdnych liniek dosiahnutých pri experimentoch smerovacím prístupmi podľa topológie.

Topology	Priemerný počet prázdnych liniek			
	Kombinovaný	Výber cesty	Predikcia metriky	Najlepší SPF
Grid25	0	2.709	0	12
SwitchBlock	0	0	1.911	10
GEANT4	0	0	0	16
20Nodes	0	1	0.624	8
30Nodes	1.501	1	4.067	11

v prípade využitia smerovacích prístupov využívajúcich strojové učenie. V niektorých prípadoch je zvýšenie málo badateľné, v iných prípadoch môže dojsť až k zvýšeniu o 87 percent.

8.3 Zhrnutie

Z výsledkov sa všeobecne dá povedať, že najlepšie výsledky boli dosahované agentom pre predikciu metriky a pri kombinácii daného agenta s agentom pre výber cesty. Kým kombinovaný prístup nie vždy dosahoval lepšie výsledky ako samostatne použitá predikcia metriky, vždy mal výsledky bližšie k lepšiemu z dvoch samostatných prístupov. To prináša výhodu v tom, že ak jeden samostatný prístup dosahuje výrazne horšie výsledky na niektorej topológii, napríklad predikcia metriky na topológii 30Nodes, pri kombinovanom prístupe síce nie sú dosiahnuté najlepšie výsledky, ale dosiahnuté výsledky sú zásadne lepšie ako pri samostatnom použití daného prístupu. To by mohlo umožňovať zlepšiť celkovú prácu smerovacieho prístupu, keďže kombinovaný prístup môže do značnej

miery potlačíť slabé miesta niektorého zo samostatných prístupov. Tým by tak tiež mohlo byť možné umožniť efektívnu prácu kombinovaného smerovacieho prístupu na väčšej množine topológií ako pri samostatnom použití jeho zložiek.

Vykonané experimenty, samozrejme, majú určité limitácie, keďže boli vykonávané na modeli siete. Pre lepšiu predstavu o práci navrhovaných smerovacích prístupov by bolo do budúcnosti vhodné experimentálne využiť dané prístupy pre smerovanie dát v simulátore počítačovej siete, alebo, ideálne, v skutočnej počítačovej sieti v laboratórnych podmienkach.

9 Záver

Táto práca si kladie za cieľ analyzovať možnosti využitia strojového učenia pre smerovanie dátových tokov v počítačových sieťach za účelom zlepšenia kvality komunikácie medzi koncovými bodmi v počítačovej sieti, identifikovať možnosti a prístupy, ako je to možné realizovať, respektíve, aké prístupy sa aktuálne skúmajú vo vedeckých publikáciách, ktoré sú spojené s optimalizáciou smerovacieho procesu v počítačových sieťach s využitím strojového učenia. Na základe vykonanej analýzy bol následne vytvorený návrh pre systém, ktorý by umožňoval centrálné riadiť počítačové siete, ktorého súčasťou by bolo možné vytvoriť smerovací modul, ktorý by využíval strojové učenie pre vykonanie smerovacích rozhodnutí. Na základe návrhu systému je vytvorený aj návrh pre samotné prístupy k smerovaniu, ktorý berie ohľad na existujúce návrhy pre smerovanie v počítačových sieťach s využitím strojového učenia, pričom je využitých viacero prístupov z analyzovaných zdrojov a taktiež je navrhnutý vlastný prístup, ktorý spája viaceré prístupy do jedného celku. Pre túto prácu bol implementovaný model počítačovej siete, na ktorom boli experimentálne overené a odskúšané navrhnuté prístupy pre smerovanie v počítačových sieťach. Nakoniec boli z experimentálne získaných dát vykonané porovnania jednotlivých navrhnutých prístupov aj s algoritmickými prístupmi využívajúcimi SPF algoritmus so staticky danými váhami liniek.

Pre vytvorenie prehľadu o existujúcich návrhoch pre riešenie optimalizácie smerovania v počítačových sieťach s využitím strojového učenia, bola vykonaná analýza existujúcich prác, ktoré sa danou problematikou zaoberajú. V týchto prácach existujú dva hlavné smery, a to využitie strojového učenia pre vykonanie samotných smerovacích rozhodnutí alebo ako podporný vstup, napríklad metrika zahltenosti siete, ktorý slúži pre podporu rozhodovania algoritmu alebo len informatívne pre administrátora počítačovej siete. Pre túto prácu sú relevantné prístupy, ktoré využívajú strojové učenie priamo pre smerovanie dát v počítačovej sieti. V tejto kategórii existuje viacero prístupov, pričom najviac relevantné prístupy sú:

- predikcia váh liniek,
- výber cesty,
- a predikcia podielov, ktoré budú využívať alternatívne cesty.

Taktiež je ešte dôležité rozdelenie na základe toho, či sa predikcia vykonáva na základe lokálneho stavu siete, to jest napríklad predikujú sa váhy linky len na základe vlastností danej linky, alebo na základe globálneho stavu siete, to jest napríklad predikujú sa váhy linky na základe celkového stavu počítačovej siete. Opis všetkých analyzovaných prác a prístupov je bližšie opísaný v kapitolách č. 4 a č. 3.

Na základe spomínanej, a vyššie opísanej, analýzy bol vytvorený návrh pre systém, ktorý by dovoľoval centrálne riadiť počítačovú sieť. Tento návrh je opísaný všeobecne a, keďže táto práca je venovaná priamo optimalizácii smerovacieho procesu s využitím strojového učenia v centrálne riadených počítačových sieťach, slúži primárne na definovanie ohraničenia potrebnej funkcionality pre smerovací proces a ohraničenia informácie, ktorú je možné využívať pre vykonanie smerovacích rozhodnutí. Celkový všeobecný opis systému je možné nájsť v sekcii č. 5.1.

Dôležitou súčasťou vyššie opísaného systému je samotný smerovací modul. V tejto práci je navrhnutých niekoľko možností prístupov k smerovaniu v počítačovej sieti s využitím strojového učenia, pričom tieto prístupy sú opísané v sekcii č. 5.2 a č. 5.3. Pre smerovanie boli vybrané nasledujúce prístupy:

- predikcia váh liniek (metrika) lokálne, každej linky samostatne, len na základe informácie o danej linke,
- predikcia váh liniek globálne, všetkých liniek spolu, na základe celkového stavu siete v matici susednosti,
- predikcia váh liniek globálne, na základe vektora informácií o každej linke,
- výber jednej cesty z množiny o n ciest s využitím algoritmu DQN,
- výber jednej cesty z množiny o n ciest s využitím algoritmu PPO,
- a predikcia váh liniek s nasledujúcim výberom z množiny o n ciest.

Algoritmy strojového učenia, ktoré sú využívané pri implementácii jednotlivých prístupov, sú opísané v sekcii č. 5.4. Opis implementácie jednotlivých prístupov je možné vidieť v sekcii č. 6.3, kde v jednotlivých podsekciiach sú opísané implementácie jednotlivých prístupov.

Spojené s návrhom a implementáciou smerovacích prístupov je návrh a implementácia prostredia, s ktorým agenti reprezentujúci jednotlivé smerovacie prístupy interagujú. Prostredie je navrhnuté v sekcii č. 5.5, kde v podsekciiach je opísané stavové prostredie, prostredie akcií, výpočet odmeny a opísaná predpokladaná interakcia agenta s prostredím. Implementácia týchto bodov súvisiacich s prostredím je spomenutá v sekcii č. 6.1. V podsekcii č. 5.5.6 sú opísané topológie, s ktorými sa v rámci tejto práce pracuje, pričom ich implementácia je opísaná v podsekcii č. 6.1.4.

Pre modelovanie fungovanie počítačovej siete bol navrhnutý model siete založený na M/M/1/K modeli fronty, tak ako je opísané v podsekcii č. 5.5.2. Opis implementácie modelu siete je možné vidieť v sekcii č. 6.2. Tento model sa využíval pri všetkých experimentoch, či už pri testovaní dosiahnutých výsledkov natrénovaných agentov využitých pri jednotlivých prístupoch, ako aj pri samotnom ich tréningu.

V kapitole č. 7 je opísaný priebeh experimentov, ktoré boli vykonané pre meranie výkonnosti navrhovaných prístupov strojového učenia pre optimalizáciu smerovania dátových tokov v počítačovej sieti. Konkrétne, v sekcii č. 7.1 je opis priebehu experimentov a spôsobu merania sledovaní kvalitatívnych vlastností komunikácie v modeli počítačovej siete. Táto práca sa pri experimentoch primárne zameriava na zníženie stratovitosti dátových paketov pri prechode komunikácie počítačovou sieťou, pričom boli vykonané aj merania na základe iných metrík, ako je oneskorenie, priemerný počet dátových tokov prechádzajúcich jednou linkou a počet prázdnych, nevyužitých liniek. V sekcii č. 7.2 je opísaný priebeh tréningu agentov pre ich využitie s niektorým z navrhovaných smerovacích prístupov. Taktiež sú v tejto sekcii spomenuté nastavené hyperparametre tréningu agentov a sú aj zaznamenané priebehy tréningu jednotlivých agentov.

Vyhodnotenie dosiahnutých výsledkov je možné vidieť v poslednej kapitole č. 8. Ako je z výsledkov očividné, najlepšie výsledky pri predikcii váh liniek dosahoval prístup, ktorý využíval len lokálnu informáciu pri svojom rozhodovaní a pričom rozhodoval samostatne o váhe každej linky. Tento prístup má aj veľkú výhodu v tom, že bez akýchkoľvek úprav je takýto prístup možné použiť na akúkoľvek počítačovú sieť, pričom vôbec nie je podstatné, aké dimenzie tvorí graf danej počítačovej siete. Ostatné ukázané prístupy, aj keď nedosahovali zlé výsledky, dosahovali všeobecne o niečo horšie výsledky ako samostatná predikcia váh liniek. Väčším obmedzením je ale nemožnosť využiť bez značných úprav takto natrénovaného agenta na ľubovoľnú počítačovú sieť, čo značne obmedzuje využiteľnosť takéhoto prístupu. Z pohľadu výberu jednej cesty z n potenciál-

ných ciest, lepšie výsledky boli dosahované agentom, ktorý bol trénovaný algoritmom DQN. O niečo horšie výsledky dosahovali agenti trénovaní algoritmom PPO. Z týchto dôvodom pre posledný prístup, v ktorom sa spája predikcia váh liniek a výber cesty z n ciest, boli zvolení agenti nasledovne:

- pre predikciu metriky bol zvolený agent, ktorý samostatne predikuje váhy jednotlivých liniek,
- pre výber cesty z n ciest bol zvolený agent, ktorý je trénovaný algoritmom DQN.

Výsledky týchto agentov pri práci samostatne a pri práci spolu sú následne porovnávané medzi sebou a s výsledkami dosiahnutými pri využití statických váh liniek spolu s algoritmom SPF. Z dosiahnutých výsledkov je možné vidieť, že všeobecne najlepšie výsledky sú dosahované agentom pre predikciu metriky a pri spoločnom použití agenta pre predikciu metriky a výber cesty, pričom pri zohľadnení stability výsledkov je najlepší prístup využívajúci oboch agentov.

Z pohľadu budúceho výskumu, mohlo by byť zaujímavé odskúšať navrhnuté prístupy pre smerovanie s využitím strojového učenia na iných, komplexnejších, modeloch a simulátoroch počítačovej siete, respektíve, priamo na reálnej počítačovej sieti v laboratórnych podmienkach. Ďalším smerom výskumu by mohlo byť zlepšenie už existujúcich navrhnutých prístupov, či už zlepšením architektúry neurónových sietí, využitím iných, viac optimálnych, hyperparametrov a aj trénovacích algoritmov, alebo zlepšením reprezentácie stavu a odmeny. Poslednou možnosťou by mohlo byť využitie prístupov, ktoré v tejto práci neboli vyskúšané, napríklad spomínaná predikcia podielov dát využívajúcich alternatívne cesty k destinácii a podobným spôsobom ich spojenie do jedného celku.

Zoznam použitej literatúry

1. HAKIRI, Akram; GOKHALE, Aniruddha; BERTHOU, Pascal; SCHMIDT, Douglas C.; GAYRAUD, Thierry. Software-Defined Networking: Challenges and research opportunities for Future Internet. *Computer Networks*. 2014, roč. 75, s. 453–471. ISSN 1389-1286. Dostupné z DOI: 10.1016/j.comnet.2014.10.015.
2. LET, G. Shine; PRATAP, C. Benin; JAGANNATH, D. J.; DOLLY, D. Raveena Judie; EVANGELINE, L. Diana. Software-Defined Networking Routing Algorithms: Issues, QoS and Models. *Wireless Personal Communications*. 2023, roč. 131, č. 3, s. 1631–1661. ISSN 1572-834X. Dostupné z DOI: 10.1007/s11277-023-10516-y.
3. *Future Intent-Based Networking: On the QoS Robust and Energy Efficient Heterogeneous Software Defined Networks*. Springer International Publishing, 2022. ISBN 9783030924355. ISSN 1876-1119. Dostupné z DOI: 10.1007/978-3-030-92435-5.
4. CLEMM, Alexander; CIAVAGLIA, Laurent; GRANVILLE, Lisandro Zambenedetti; TANTSURA, Jeff. *Intent-Based Networking - Concepts and Definitions* [RFC 9315]. RFC Editor, 2022. Request for Comments, č. 9315. Dostupné z DOI: 10.17487/RFC9315.
5. LEIVADEAS, Aris; FALKNER, Matthias. A Survey on Intent-Based Networking. *IEEE Communications Surveys and Tutorials*. 2023, roč. 25, č. 1, s. 625–655. ISSN 2373-745X. Dostupné z DOI: 10.1109/comst.2022.3215919.
6. BEHRINGER, Michael H.; PRITIKIN, Max; BJARNASON, Steinthor; CLEMM, Alexander; CARPENTER, Brian E.; JIANG, Sheng; CIAVAGLIA, Laurent. *Autonomic Networking: Definitions and Design Goals* [RFC 7575]. RFC Editor, 2015. Request for Comments, č. 7575. Dostupné z DOI: 10.17487/RFC7575.
7. MOY, J. *OSPF Version 2*. 1994. Dostupné z DOI: 10.17487/rfc1583.
8. *OSI IS-IS Intra-domain Routing Protocol*. 1990. Dostupné z DOI: 10.17487/rfc1142.

9. SAVAGE, D.; NG, J.; MOORE, S.; SLICE, D.; PALUCH, P.; WHITE, R. *Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)*. 2016. Dostupné z DOI: 10.17487/rfc7868.
10. MALKIN, G. *RIP Version 2*. 1998. Dostupné z DOI: 10.17487/rfc2453.
11. GUCK, Jochen W.; VAN BEMTEN, Amaury; REISSLEIN, Martin; KELLERER, Wolfgang. Unicast QoS Routing Algorithms for SDN: A Comprehensive Survey and Performance Evaluation. *IEEE Communications Surveys and Tutorials*. 2018, roč. 20, č. 1, s. 388–415. ISSN 1553-877X. Dostupné z DOI: 10.1109/comst.2017.2749760.
12. AMIN, Rashid; ROJAS, Elisa; AQDUS, Aqsa; RAMZAN, Sadia; CASILLAS-PEREZ, David; ARCO, Jose M. A Survey on Machine Learning Techniques for Routing Optimization in SDN. *IEEE Access*. 2021, roč. 9, s. 104582–104611. ISSN 2169-3536. Dostupné z DOI: 10.1109/access.2021.3099092.
13. KIM, Gyungmin; KIM, Yohan; LIM, Hyuk. Deep Reinforcement Learning-Based Routing on Software-Defined Networks. *IEEE Access*. 2022, roč. 10, s. 18121–18133. ISSN 2169-3536. Dostupné z DOI: 10.1109/access.2022.3151081.
14. SUN, Penghao; GUO, Zehua; LI, Junfei; XU, Yang; LAN, Julong; HU, Yuxiang. Enabling Scalable Routing in Software-Defined Networks With Deep Reinforcement Learning on Critical Nodes. *IEEE/ACM Transactions on Networking*. 2022, roč. 30, č. 2, s. 629–640. ISSN 1558-2566. Dostupné z DOI: 10.1109/tnet.2021.3126933.
15. ALMASAN, Paul; SUÁREZ-VARELA, José; RUSEK, Krzysztof; BARLET-ROS, Pere; CABELLOS-APARICIO, Albert. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *Computer Communications*. 2022, roč. 196, s. 184–194. ISSN 0140-3664. Dostupné z DOI: 10.1016/j.comcom.2022.09.029.
16. WEI, Huihong; ZHAO, Yi; XU, Ke. G-Routing: Graph Neural Networks-Based Flexible Online Routing. *IEEE Network*. 2023, roč. 37, č. 4, s. 90–96. ISSN 1558-156X. Dostupné z DOI: 10.1109/mnet.012.2300052.
17. BHAVANASI, Sai Shreyas; PAPPONE, Lorenzo; ESPOSITO, Flavio. Dealing With Changes: Resilient Routing via Graph Neural Networks and Multi-Agent Deep Reinforcement Learning. *IEEE Transactions on Network and Service Management*. 2023, roč. 20, č. 3, s. 2283–2294. ISSN 2373-7379. Dostupné z DOI: 10.1109/tnsm.2023.3287936.

18. HE, Yuanhao; XIAO, Geyang; LIANG, Yuan; LU, Donghui; CHENG, Xiaofeng. A Graph Reinforcement Learning Routing Scheme Based on Nodes and Edges Co-Embedding. In: *ICC 2024 - IEEE International Conference on Communications*. IEEE, 2024, s. 5431–5436. Dostupné z doi: 10.1109/icc51166.2024.10623011.
19. AGUIRRE SANCHEZ, Lizeth Patricia; SHEN, Yao; GUO, Minyi. DQS: A QoS-driven routing optimization approach in SDN using deep reinforcement learning. *Journal of Parallel and Distributed Computing*. 2024, roč. 188, s. 104851. ISSN 0743-7315. Dostupné z doi: 10.1016/j.jpdc.2024.104851.
20. AGUIRRE SANCHEZ, Lizeth Patricia; SHEN, Yao; GUO, Minyi. MDQ: A QoS-Congestion Aware Deep Reinforcement Learning Approach for Multi-Path Routing in SDN. *Journal of Network and Computer Applications*. 2025, roč. 235, s. 104082. ISSN 1084-8045. Dostupné z doi: 10.1016/j.jnca.2024.104082.
21. KURTULUS, Baris; KARAKUS, Murat; GULER, Evrim. DQRL: A Directed Acyclic Graph and RL-Based Framework for QoS-Centric Routing in Multi SDNs. In: *2025 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. IEEE, 2025, s. 1–6. Dostupné z doi: 10.1109/blackseacom65655.2025.11193937.
22. WU, Jiawei; ZHU, Zelin. Intelligent routing optimization for SDN based on PPO and GNN. *Journal of Network and Computer Applications*. 2025, roč. 242, s. 104249. ISSN 1084-8045. Dostupné z doi: 10.1016/j.jnca.2025.104249.
23. LIU, Chenyi; WU, Pingfei; XU, Mingwei; YANG, Yuan; GENG, Nan. Scalable Deep Reinforcement Learning-Based Online Routing for Multi-Type Service Requirements. *IEEE Transactions on Parallel and Distributed Systems*. 2023, roč. 34, č. 8, s. 2337–2351. ISSN 2161-9883. Dostupné z doi: 10.1109/tpds.2023.3284651.
24. HUANG, Linqiang; YE, Miao; XUE, Xingsi; WANG, Yong; QIU, Hongbing; DENG, Xiaofang. Intelligent routing method based on Dueling DQN reinforcement learning and network traffic state prediction in SDN. *Wireless Networks*. 2022, roč. 30, č. 5, s. 4507–4525. ISSN 1572-8196. Dostupné z doi: 10.1007/s11276-022-03066-x.
25. NAZARI, Hamed; SHAHRIAR, Nashid; BORYŁO, Piotr. MARINA: Multi-Agent Reinforcement Learning-Based Routing with Intelligent Network Adaptation in SDN. In: *2025 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2025, s. 631–637. Dostupné z doi: 10.1109/ccece64018.2025.11364348.

26. DUGAEV, Dmitrii A.; MATVEEV, Ivan G.; SIEMENS, Eduard; SHUVALOV, Viatcheslav P. Adaptive Reinforcement Learning-Based Routing Protocol for Wireless Multihop Networks. In: *2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*. IEEE, 2018, s. 209–218. Dostupné z doi: 10.1109/apieie.2018.8545412.
27. MARTIN, Ignacio; TROIA, Sebastian; HERNANDEZ, Jose Alberto; RODRIGUEZ, Alberto; MUSUMECI, Francesco; MAIER, Guido; ALVIZU, Rodolfo; GONZALEZ DE DIOS, Oscar. Machine Learning-Based Routing and Wavelength Assignment in Software-Defined Optical Networks. *IEEE Transactions on Network and Service Management*. 2019, roč. 16, č. 3, s. 871–883. ISSN 2373-7379. Dostupné z doi: 10.1109/tnsm.2019.2927867.
28. SANTOS, Carlos Lester Duenas; MEZHER, Ahmad Mohamad; LEÓN, Juan Pablo Astudillo; BARRERA, Julian Cardenas; GUERRA, Eduardo Castillo; MENG, Julian. ML-RPL: Machine Learning-Based Routing Protocol for Wireless Smart Grid Networks. *IEEE Access*. 2023, roč. 11, s. 57401–57414. ISSN 2169-3536. Dostupné z doi: 10.1109/access.2023.3283208.
29. ANDREOLETTI, Davide; ROTTONDI, Cristina; GIORDANO, Silvia; BIANCO, Andrea. Scalable and Privacy-Preserving Inter-AS Routing Through Machine-Learning-Based Graph Pruning. *IEEE Access*. 2025, roč. 13, s. 21891–21905. ISSN 2169-3536. Dostupné z doi: 10.1109/access.2025.3536545.
30. CICIOĞLU, Murtaza; ÇALHAN, Ali. MLaR: machine-learning-assisted centralized link-state routing in software-defined-based wireless networks. *Neural Computing and Applications*. 2022, roč. 35, č. 7, s. 5409–5420. ISSN 1433-3058. Dostupné z doi: 10.1007/s00521-022-07993-w.
31. LI, Ming; WANG, Yuewen; WANG, Zhaowen; ZHENG, Huiying. A deep learning method based on an attention mechanism for wireless network traffic prediction. *Ad Hoc Networks*. 2020, roč. 107, s. 102258. ISSN 1570-8705. Dostupné z doi: 10.1016/j.adhoc.2020.102258.
32. BI, Jing; ZHANG, Xiang; YUAN, Haitao; ZHANG, Jia; ZHOU, MengChu. A Hybrid Prediction Method for Realistic Network Traffic With Temporal Convolutional Network and LSTM. *IEEE Transactions on Automation Science and Engineering*. 2022, roč. 19, č. 3, s. 1869–1879. ISSN 1558-3783. Dostupné z doi: 10.1109/tase.2021.3077537.
33. XIE, Ruotian; WEN, Jigang; CHEN, Xiaodi; XIE, Kun; LIANG, Wei; XIONG, Naixue; ZHANG, Dafang; XIE, Gaogang; LI, Kenli. M²STL: Multi-Range

- Multi-Level Spatial-Temporal Learning Model for Network Traffic Prediction. *IEEE Transactions on Network Science and Engineering*. 2024, roč. 11, č. 5, s. 4315–4329. ISSN 2334-329X. Dostupné z doi: 10.1109/tNSE.2024.3417371.
34. HUO, Yonghua; LIU, Ying; ZHAO, Shilong; YU, Peng. Leveraging Graph Attention Network for SDN Routing Performance Prediction. In: *2022 International Conference on Information Processing and Network Provisioning (ICIPNP)*. IEEE, 2022, s. 15–19. Dostupné z doi: 10.1109/icipnp57450.2022.00010.
35. RUSEK, Krzysztof; SUAREZ-VARELA, Jose; ALMASAN, Paul; BARLET-ROS, Pere; CABELLOS-APARICIO, Albert. RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN. *IEEE Journal on Selected Areas in Communications*. 2020, roč. 38, č. 10, s. 2260–2270. ISSN 1558-0008. Dostupné z doi: 10.1109/jsac.2020.3000405.
36. FERRIOL-GALMÉS, Miquel; PAILLISSE, Jordi; SUÁREZ-VARELA, José; RUSEK, Krzysztof; XIAO, Shihan; SHI, Xiang; CHENG, Xiang; BARLET-ROS, Pere; CABELLOS-APARICIO, Albert. RouteNet-Fermi: Network Modeling With Graph Neural Networks. *IEEE/ACM Transactions on Networking*. 2023, roč. 31, č. 6, s. 3080–3095. ISSN 1558-2566. Dostupné z doi: 10.1109/tnet.2023.3269983.
37. KAJA, Siddardha; SHAKSHUKI, Elhadi; MALIK, Haroon; YASAR, Ansar. One Hop Routing Optimization Approach Using Machine Learning. *Procedia Computer Science*. 2024, roč. 241, s. 85–96. ISSN 1877-0509. Dostupné z doi: 10.1016/j.procs.2024.08.014.
38. FUJIMOTO, Scott; HOOFF, Herke van; MEGER, David. *Addressing Function Approximation Error in Actor-Critic Methods*. arXiv, 2018. Dostupné z doi: 10.48550/ARXIV.1802.09477.
39. HESSEL, Matteo; MODAYIL, Joseph; HASSELT, Hado van; SCHAUL, Tom; OSTROVSKI, Georg; DABNEY, Will; HORGAN, Dan; PIOT, Bilal; AZAR, Mohammad; SILVER, David. *Rainbow: Combining Improvements in Deep Reinforcement Learning*. arXiv, 2017. Dostupné z doi: 10.48550/ARXIV.1710.02298.
40. MNIH, Volodymyr; KAVUKCUOGLU, Koray; SILVER, David; RUSU, Andrei A.; VENESS, Joel; BELLEMARE, Marc G.; GRAVES, Alex; RIEDMILLER, Martin; FIDJELAND, Andreas K.; OSTROVSKI, Georg; PETERSEN,

- Stig; BEATTIE, Charles; SADIK, Amir; ANTONOGLOU, Ioannis; KING, Helen; KUMARAN, Dharshan; WIERSTRA, Daan; LEGG, Shane; HASSABIS, Demis. Human-level control through deep reinforcement learning. *Nature*. 2015, roč. 518, č. 7540, s. 529–533. ISSN 1476-4687. Dostupné z DOI: 10.1038/nature14236.
41. SUTTON, Richard S.; BARTO, Andrew G. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. Dostupné tiež z: <http://incompleteideas.net/book/the-book-2nd.html>.
42. SCHULMAN, John; WOLSKI, Filip; DHARIWAL, Prafulla; RADFORD, Alec; KLIMOV, Oleg. *Proximal Policy Optimization Algorithms*. arXiv, 2017. Dostupné z DOI: 10.48550/ARXIV.1707.06347.
43. SILVER, David; LEVER, Guy; HEES, Nicolas Manfred Otto; DEGRIS, Thomas; WIERSTRA, Daan; RIEDMILLER, Martin A. Deterministic Policy Gradient Algorithms. In: *International Conference on Machine Learning*. 2014. Dostupné tiež z: <https://api.semanticscholar.org/CorpusID:13928442>.
44. FORTUNATO, Meire; AZAR, Mohammad Gheshlaghi; PIOT, Bilal; MENICK, Jacob; OSBAND, Ian; GRAVES, Alex; MNIH, Vlad; MUNOS, Remi; HASSABIS, Demis; PIETQUIN, Olivier; BLUNDELL, Charles; LEGG, Shane. *Noisy Networks for Exploration*. arXiv, 2017. Dostupné z DOI: 10.48550/ARXIV.1706.10295.
45. SCHAUL, Tom; QUAN, John; ANTONOGLOU, Ioannis; SILVER, David. Prioritized Experience Replay. 2015. Dostupné z DOI: 10.48550/ARXIV.1511.05952.
46. HOLLENSTEIN, Jakob; AUDDY, Sayantan; SAVERIANO, Matteo; RENAUDO, Erwan; PIATER, Justus. Action Noise in Off-Policy Deep Reinforcement Learning: Impact on Exploration and Performance. 2022. Dostupné z DOI: 10.48550/ARXIV.2206.03787.

Zoznam skratiek

CNN Convolutional Neural Network.

DDPG Deep Deterministic Policy Gradient.

DNN Deep Neural Networks.

DQN Deep Q Network.

DRL Deep Reinforcement Learning.

ECMP Equal-Cost Multi-Pathing.

EIGRP Enhanced Interior Gateway Routing Protocol.

GAN Graph Attention Network.

GCN Graph Convolutional Network.

GNN Graph Neural Network.

IBN Intent-Based Networking.

LSTM Long Short-Term Memory.

MANET Mobile Ad hoc Network.

MDP Markov Decision-making Process.

OSPF Open Shortest Path First.

PPO Proximal Policy Optimization.

ReLU Rectified Linear Unit.

RIP Routing Information Protocol.

RL Reinforcement Learning.

RPL Routing Protocol for Low-Power and Lossy Networks.

RWA Routing and Wavelength Assignment.

SDN Software-Defined Networking.

SPF Shortest Path First.

SVM Support Vector Machine.

TCN Temporal Convolutional Network.

TD3 Twin Delayed Deep Deterministic Policy Gradient.

TRPO Trust Region Policy Optimization.

Zoznam príloh

Príloha A Používateľská príručka

Príloha B CD médium – záverečná práca v elektronickej podobe,

A Používateľská príručka

Táto sekcia je venovaná opisu lokálneho spustenia experimentov opísaných a realizovaných v tejto práci. Zdrojový kód, ktorý je potrebný pre spustenie opísaných experimentov, je súčasťou Príloha B.

Pri zdrojovom kóde v súbore *requirements.txt* sú spísané knižnice použité pri realizácii projektu a pre spustenie všetkých experimentov. Experimenty sa spúšťajú cez skript *main.py*, pričom pomocou argumentov príkazového riadku je možné určiť, aký experiment má byť realizovaný. Podporované argumenty príkazového riadku sú nasledovné:

- *-a <agent>* – opisuje, aký agent má byť použitý v experimente. Podporované argumenty sú *metric*, *metric_all*, *path*, *path_ppo* a *comb*.
- *-t <0-5>* – stanovuje, na ktorej topológii má prebiehať tréning agenta. Podporované sú hodnoty 0 až 5. Nie je relevantné pri generickom tréningu.
- *-e* – stanovuje, či má prebiehať len experimentálne testovanie už natrénovaného agenta. V tomto prípade musí byť manuálne v skripte nastavené ako má prebiehať test, to jest, kde je uložený agent a aký to je vlastne agent.
- *-v <v_>* – v prípade predikcie metriky súčasne pre všetky linky, stanovuje využitie reprezentácie stavu za pomoci matice susednosti (*v1*) alebo reprezentácie stavu pomocou vektora vlastností liniek (*v2*).
- *-g* – stanovuje, či má prebiehať tréning genericky.

Vlastnosti tréningu a testovania agentov, ako aj ich hyperparametre, je možné nastaviť v dátových štruktúrach v súbore *configurations.py*, ktorého cesta je *./src/configurations.py*. V týchto dátových štruktúrach sú opísané vlastnosti topológií (v zoznamoch *topologies* a *topologies_index*), parametre tréningu ako dĺžka tréningu, počet krokov epizódy, prednastavená topológia, počet ciest a hodnoty seed-ov (v *train_parameters*), parametre tréningu metriky (v *metric_parameters*), parametre tréningu výberu cesty s DQN (v *path_parameters*) a parametre tréningu výberu cesty s PPO (v *path_parameters_ppo*). Taktiež je tomto mieste možné

nastaviť agenta, ktorý má byť načítaný z pamäti pri využití kombinovanej metódy.